Universidade de Aveiro Departamento de Engenharia Mecânica
2013

João José Poito Coelho

Sistemas distribuídos para a monitorização em ambiente industrial

João José Poito Coelho

Sistemas distribuídos para a monitorização em ambiente industrial

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Mecânica, realizada sob orientação científica de José Paulo Oliveira Santos, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro.

O júri / The jury

Presidente / President

Prof. Doutor Vitor Manuel Ferreira dos Santos
Professor Associado do Departamento de Engenharia Mecânica da Universidade de Aveiro

Vogais / Committee

Prof. Doutor José Paulo Oliveira Santos
Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro (orientador)

Prof. Doutor Pedro Nicolau Faria da Fonseca
Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Agradecimentos / Acknowledgements

Agradeço ao meu Orientador, Professor Doutor José Paulo Santos, a possibilidade de poder realizar este trabalho e pela sua disponibilidade prestada.
Agradeço à Bresimar, ao Eng. Ricardo Carvalho, ao Eng. Alexandre Ferreira e especialmente Eng. Armando Cavaleiro, que acompanhou o meu trabalho mais de perto, pela oportunidade de desenvolver este trabalho, em que me foi dado todo o material e apoio necessário para a realização de um bom trabalho.
Agradeço a todos os meus amigos, pela amizade e apoio que me demonstraram ao longo deste trabalho. Em especial quero agradecer a todos os amigos mais chegados, em que foram passados momentos inesquecíveis de grande companheirismo ao longo do curso.
Por último, agradeço especialmente à minha Mãe, à minha irmã, ao Manel, à Helena e ao Zé pelo apoio constante, ao longo desta longa caminhada, muitas vezes difícil, mas que foi ultrapassada com sucesso.

Palavras-chave

Barramento de campo, CAN-*Controller Area Network*, CANopen, CIM-*Computer Integrated Manufacturing*, Comunicações tempo-real, Microcontroladores, Sistemas de controlo distribuídos em tempo-real, Sistemas embebidos, Sistemas de manufactura.

Resumo

Nas últimas décadas tem sido visível a proliferação dos sistemas incorporados ou embebidos nas aplicações industriais. Nestes sistemas, o controlo é geralmente distribuído, o que implica que vários componentes troquem informação entre si. Por outro lado, essa troca de informação é suportada por uma rede de comunicação que interliga esses mesmos. Devido a rigorosos requisitos temporais, aspectos como o protocolo de acesso ao meio, a velocidade de transmissão, o tamanho das mensagens, serviços de controlo e supervisão da rede apresentam-se de elevada relevância para este tipo de sistemas.

Na presente dissertação é revelado um estudo de importantes fatores tecnológicos em ambiente CIM e apresentada uma solução de baixo custo a ser implementada no chão de fábrica, ao nível do controlo de sensores e atuadores. Nesse intuito, foram estudados um protocolo aberto e uma pilha protocolar *open source* que permitisse escalabilidade, flexibilidade e mecanismos de deteção de erros.

Para comprovar o funcionamento do sistema foi implementado um demonstrador representativo com o protocolo de comunicação proposto, o CANopen. Este é composto por dois nós CANopen e por um computador utilizado para analisar e configurar a rede.

Keywords

Fieldbus, CAN-*Controller Area Network*, CANopen, CIM-*Computer Integrated Manufacturing*, Real-time communications, Microcontrollers, Real-time distributed control systems, Embedded systems, Manufacturing systems

Abstract

In the last decades the proliferation of embedded systems has been visible in industrial application. The control in these systems is generally distributed which implies an exchange of information between several components. Subsequently, the information exchange is supported by a communication network that connects all the respective components. Due to rigorous time requirements, features such the medium access protocol, bitrate, message length, control services and network supervision present relevance to the distributed systems.

The current dissertation presents a study in important technological factors in CIM environment and a low cost solution to be implemented at shop floor, in sensors and actuators control level. In that sense, it was studied an open protocol and a open source stack which allowed modularity, scalability, flexibility and error detection mechanisms.

To verify the system's performance, a representative demonstrator containing the proposed communication protocol was implemented. This is constituted with two CANopen nodes and a computer to analyze and configure the network.

Conteúdo

Lista de Tabelas

Lista de Figuras

Lista de Acrónimos

ADC *Analog-to-Digital Converter*

ADU *Application Data Unit*

AGV *Automatic Guided Vehicle*

AS/RS *Automated Storage/Retrieval System*

CAD *Computer Aided Design*

CAE *Computer Aided Engineer*

CAM *Computer Aided Manufacturing*

CAN *Controller Area Network*

CAP *Computer Aided Planning*

CAQ *Computer Aided Quality*

CiA *CAN in Automation*

CIM *Computer Integrated Manufacturing*

CIP *Common Industrial Protocol*

CNC Controlo Numérico Computorizado

COB-ID *Communication Object Identifier*

CRC *Cyclic Redundancy Check*

CSMA/NBA *Carrier Sense Multiple Access with Non-destructive Bitwise Arbitration*

DLL *Dynamic Link Library*

DNC *Direct Numerical Control*

DS *Draft Standard*

DR *Draft Recommendation*

EDS *Electronic Data Sheet*

EEPROM *Electrically Erasable Programmable Read-Only Memory*

ET *Event-Triggered*

EtherCAT *Ethernet Control Automation Tecnhonogy*

FIFO *First In, First Out*

HMI *Human Machine Interface*

I/O *Input/Output*

ISO *International Organization for Standardization*

JIT *Just in Time*

LCD *Liquid Crystal Display*

LED *Light Emitting Diode*

MAC *Medium Access Control*

NMT *Network Management*

OSI *Open Systems Interconnection*

PDO *Process Data Objects*

PDU *Protocol Data Unit*

PNO *Profibus User Organization*

PPC *Production Planning and Control*

PROFIBUS *PROcess Field BUS*

PROFINET *PROcess Field NET*

PWM *Pulse Width Modulation*

RAM *Random Access Memory*

ROM *Read Only Memory*

SCADA *Supervisory Control and Data Acquisition*

SDO *Service Data Objects*

SPI *Serial Peripheral Interface*

TDMA *Time Division Multiple Access*

TCP/IP *Transmission Control Protocol/Internet Protocol*

TT *Time-Triggered*

Tq *Time-quantum*

USB *Universal Serial Bus*

WIP *Work in Process*

XML *eXtensible Markup Language*

Capítulo 1

Introdução

1.1 Enquadramento e Motivação

Até meados do anos 70, os sistemas de controlo automático computorizado baseavam-se em arquiteturas centralizadas, que consistiam num computador a executar um ciclo infinito de um algoritmo de controlo que aplicava determinadas atuações a partir de dados sensoriais. Contudo, já na altura, eram indicados certos inconvenientes como a elevada cablagem e o elevado esforço computacional, uma vez que todos os dispositivos estavam ligados a uma unidade central de processamento [1].

A pressão económica e a legislação laboral e ambiental mais restritiva conduziram a que a indústria recorresse a sistemas com melhores níveis de automatização numa perspectiva de melhorar o controlo de qualidade de processos e produtos e também de reduzir custos. Com o passar do tempo os problemas das arquiteturas centralizadas tornaram-se cada vez mais evidentes. Por essa razão, surgiram os sistemas de controlo distribuídos, uma solução que se baseia na distribuição de funcionalidades do sistema por várias unidades processadoras interligadas entre si por uma rede de comunicação [1].

O aumento da integração eletrónica permitiu construir nós com dimensões reduzidas, com interface *Input/Output* (I/O) e uma ou mais interfaces de comunicação. As reduzidas dimensões permitiram que estas pudessem ser colocadas dentro dos próprios sistemas a controlar, dando origem aos sistemas de controlo distribuídos e embebidos. Este processo de implementar um sistema de controlo no interior do próprio sistema a controlar, surge em aplicações desde o controlo de um automóvel, de um avião, de máquinas ferramentas, de robôs, de instrumentação médica, ou até mesmo ao controlo de experiências em laboratórios.

Os sistemas de controlo distribuído acarretam vantagens como a diminuição dos custos de cablagem, melhor utilização de recursos, composição dos sistemas por módulos, facilidade de reconfiguração e diagnóstico. Contudo, implica que vários componentes troquem informação entre si. Essa informação é geralmente trocada recorrendo a uma rede de comunicação que interliga todos os componentes intervenientes no sistema de controlo. Este tipo de implementação pode por isso trazer problemas como o aparecimento de atrasos introduzidos na malha pelo processo de comunicação, o que degrada o desempenho do sistema.

1.2 Problema

A crescente competitividade e globalização do mercado obrigam a novos desafios e novos conceitos para os sistemas de produção, com vista ao aumento da sua flexibilidade, modularidade, adaptabilidade, capacidade de cooperação e de integração. A integração dos recursos fabris, o suporte dos fluxos de informação na instalação fabril como forma de aumentar a produtividade e a flexibilidade são elementos essenciais para o aumento da competitividade das empresas e a chave para o seu sucesso. Todavia, a grande diversidade de protocolos de comunicação e de equipamentos fabris dificultam essa integração e aumentam os custos de produção das empresas.

As comunicações industriais têm requisitos diferentes para os vários níveis do processo industrial. As aplicações industriais usadas no chão da fábrica contêm cada vez mais elementos com sistemas incorporados ou embebidos. No nível em que este trabalho incide, um dos requisitos principais passa pela sua precisão temporal, uma vez que este tipo de redes geralmente são dimensionados para partilhar dados numa rede de comunicação, com um número de nós significativo, que por vezes controlam processos críticos, onde falhas podem resultar em consequências imensuráveis. No desenvolvimento de sistemas distribuídos é fundamental apresentarem capacidades como a rápida adaptação às mudanças, maior robustez e a fácil integração para contrariar principalmente a rigidez de algumas arquiteturas de controlo sem a capacidade de resposta dinâmica.

O controlo implementado de uma forma distribuída, poderá acarretar atrasos associados à comunicação da informação de controlo, devido à transmissão de dados, a erros e omissões. A questão sobre a precisão temporal associada a uma rede partilhada é difícil de ser assegurada quando se trata de transmitir informação de controlo neste tipo de redes. O protocolo de acesso ao meio, a velocidade de transmissão, o tamanho da mensagem e quantidade de tráfego na rede são condicionantes, com um peso relevante no atraso temporal entre o instante em que é amostrado um processo e o instante em que é aplicado o controlo nesse mesmo processo.

Tendo em conta que os atrasos podem ser influenciados pelo protocolo de comunicação utilizado e pela sobrecarga da rede, nos sistemas de tempo-real, estes aspectos ganham maior importância, uma vez que se não forem satisfeitas as suas necessidades de precisão temporal, quer nos cálculos quer nas comunicações, a qualidade do desempenho do controlo do sistema é comprometida. Por exemplo, um sistema que tenha a função de controlar a posição de um braço robótico, certas ordens terão de ser dadas a tempo, para evitar consequências com custos imensuráveis.

1.3 Objetivos

No âmbito desta dissertação, serão estudados importantes fatores tecnológicos para desenvolver uma solução particular para uma implementação em ambiente *Computer Integrated Manufacturing* (CIM) para os sistemas integrados de produção. O objetivo principal passa pela implementação de uma rede industrial, para interligar dispositivos I/O, que seja fiável, de baixo custo, que não esteja dependente do seu posicionamento físico e capaz de ser reconfigurado sem necessitar de ser reprogramado.

Nesse sentido, os seguintes objetivos deverão ser realizados:

- Levantamento do estado de arte: Em que serão abordados aspetos importantes sobre o modelo CIM e algumas redes de comunicação em ambiente industrial;
- Definir uma rede de campo após realizadas algumas comparações;
- Implementar os módulos I/O com um protocolo de comunicação escolhido.
- Desenvolver uma interface que permita que o utilizador seja capaz de efetuar diferentes configurações;

A fim de atingir os níveis desejados de flexibilidade e agilidade, a implementação deve ser capaz de cumprir os seguintes requisitos:

- Modularidade: A atribuição de diferentes funções a módulos específicos torna o sistema especializado e escalável;
- Reduzir o esforço de programação: O ato de reprogramar sistemas inteiros em ordem para lidar com as mudanças de produção ou a adição ou remoção de componentes pode ser muito demorada e dispendiosa. Assim, um dos principais objetivos é ter um sistema com pouca ou nenhuma necessidade de reprogramação, trocando todo o trabalho para configuração, no qual será mais fácil e mais rápido;
- Possua capacidades para interagir com outros protocolos mais utilizado para níveis superiores.

1.4 Organização da dissertação

A dissertação está organizada em 5 capítulos:

- No capítulo 2 é apresentado o estado de arte que aborda os sistemas de manufatura e o paradigma CIM. Em virtude da abordagem à produção em ambiente CIM são descritas características gerais sobre as redes de comunicação industrias e os sistemas de controlo.

- No capítulo 3 são apresentados alguns protocolos de comunicação industriais em que são mencionadas as principais características gerais de cada um. O protocolo CANopen é descrito em mais pormenor, uma vez que serviu de suporte na realização deste trabalho.

- No capítulo 4 é justificada as escolhas realizadas na implementação e descreve em detalhe alguns pormenores sobre os nós criados e da interface desenvolvida para a configuração dos respetivos nós.

- No capítulo 5 são apresentadas as conclusões relativas ao trabalho desenvolvido e algumas sugestões para trabalho futuro.

1.5 Publicações cientificas

De seguida serão apresentados alguns estudos desenvolvidos no âmbito de dissertações que compõem o estado de arte dos sistemas de controlo distribuídos.

O autor [1] apresenta uma contribuição teórica nas redes de comunicação em sistemas de controlo distribuídos, impulsionada por estes sistemas possuírem elevada latência quando comparada com a correspondente latência num sistema não distribuído, no qual é induzida pela rede e poderá causar degradação do desempenho e estabilidade do controlo. O seu estudo é focado nas redes de comunicação baseadas em barramento elétrico série com meio partilhado, nos aspectos da arquitetura funcional, nos paradigmas de disparo das comunicações e nos protocolos *Medium Access Control* (MAC). Através de uma ferramenta de simulação *TrueTime*, o autor simulou modelos simples de sistemas distribuídos de controlo, que permitiram analisar o efeito de latência induzido pela rede, sob influência de vários padrões de tráfego e de vários protocolos de acesso ao meio MAC.

O autor [2] na procura de uma solução inovadora, versátil, económica, de alto desempenho e fácil utilização no âmbito da domótica descreve a implementação de uma rede domótica baseada nas redes Ethernet e CANopen. A sua implementação passa por utilizar uma rede *Controller Area Network* (CAN)/CANopen como barramento de campo para ligar sensores a uma rede central, concebendo um pequeno sistema de iluminação utilizando sensores e atuadores usados em Domótica e o *gateway* CANopen-Ethernet implementado. Descreve alguns dos sensores utilizados, como:

- *Binary Input*;
- *Binary Output*;
- *Analog Input*;
- *Analog Output*;
- *Dimmer*;
- Detetor de Movimento;
- Detetor de Presença;
- Painéis de Controlo;
- Detetores de Humidade;
- Sensor de Luz;
- Sensores de Temperatura;
- Controlo de Temperatura;
- Medidor de Consumo Elétrico;
- *Orbiters*.

Como trabalho futuro faz referência ao porte da pilha protocolar CANopen utilizada no computador a partir do "CAN Festival"para os microcontroladores.

O autor [3] analisou a interligação de uma rede do tipo *Fieldbus*, vocacionada para o nível mais baixo da hierarquia CIM, numa infraestrutura global de comunicações industriais. O objetivo do autor passou por permitir a integração, em ambiente CIM, das funcionalidades e atividades cobertas por esse tipo de redes, que são especialmente vocacionadas para aplicações de tempo crítico. Apresenta um estudo realizado no âmbito da produção integrada por computador, os fluxos de informação em ambiente CIM e redes de comunicação para os níveis inferiores da hierarquia CIM.

O autor [4] apresenta um trabalho que tem por base um sistema com um nó controlador responsável por receber informação do nó sensor, processar essa informação e apresentar, em tempo útil, uma determinada informação ao nó atuador, para que este

tenha um determinado comportamento. O nó controlador neste trabalho usa técnicas de controlo digital, nomeadamente controlo adaptativo, sistema capaz de modificar o seu comportamento em resposta a variações dinâmicas do processo a controlar ou a variações do ponto de funcionamento do processo, implementado num microcontrolador de baixo custo, tendo como suporte o barramento CAN. Apresenta um estudo benéfico para o levantamento realizado sobre o protocolo CAN.

Os autores [5] apresentam um artigo útil para a introdução ao estudo dos sistemas de produção. Apresenta um estudo sobre os paradigmas de manufatura, tipos de produção, requisitos futuros dos sistemas de produção, arquiteturas clássicas para sistemas de controlo e arquiteturas de uma célula de produção.

Capítulo 2

Estado da Arte

A produção consiste no processo de transformação que converte matéria-prima ou produtos semi-acabados em produtos finais com valor no mercado. Os produtos são transformados a partir do trabalho manual, das máquinas, ferramentas e energia. Este processo pode ser decomposto em vários passos até ser obtido o produto final. Os passos individuais são definidos como operações de produção [6].

As empresas ilustradas podem ser divididas em dois tipos. Empresas de manufatura que são tipicamente identificadas pela produção de artigos discretos, como o caso dos automóveis, computadores, máquinas-ferramentas e o componentes que as integram. Empresas de processamento que estão relacionadas com um processo de produção contínua como o caso da produção de energia, papel, comida, materiais de construção, químicos e plásticos [7].

As estratégias organizacionais que definem as diretrizes de produção, tais como, o tipo de produção e o plano de produção a longo ou médio prazo, são influenciadas principalmente pela procura e pelas perturbações externas (ver figura 2.1).

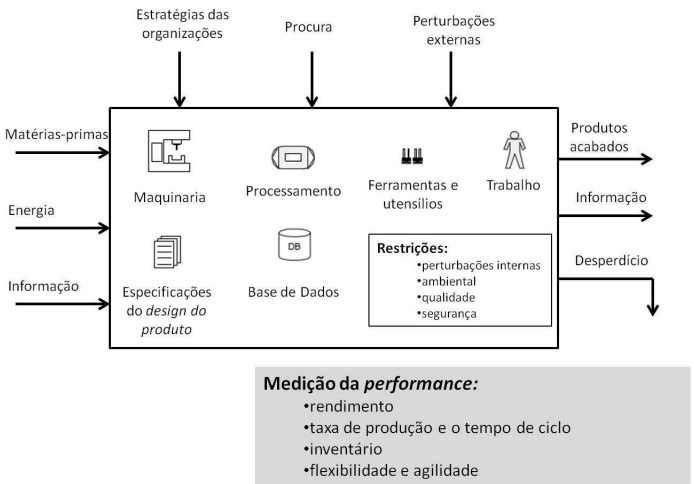

Figura 2.1: Modelo abstrato do sistema de manufatura (Adaptado a partir de [8]).

Os resultados do processo de produção são os produtos acabados que serão entregues ao mercado de acordo com as exigências dos cliente. Tendo em conta a vida útil de um produto, devido à variação na procura e das perturbações externas, um bom sistema

de planeamento e controlo, que passa pela possibilidade de serem apresentadas ações corretivas ao longo do tempo, pode ser uma aspeto fulcral para o sucesso de uma empresa.

2.1 História da evolução dos paradigmas de manufatura

O ambiente fabril encontra-se em constante adaptação às exigências dos clientes, inovando desta forma com a informatização e tecnologias de automação (ver figura 2.2 e 2.3).

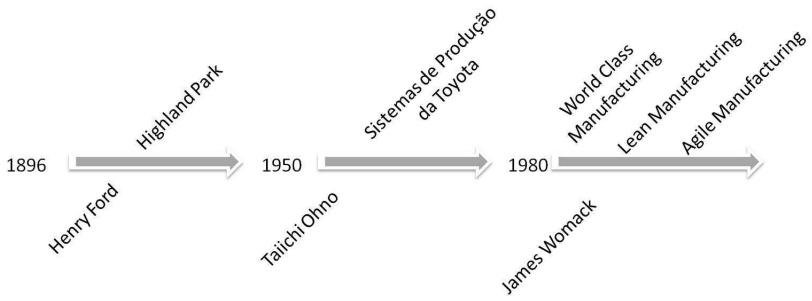

Figura 2.2: Evolução cronográfica dos paradigmas de produção.

Antes do século XX, predominava a produção do tipo artesanal, em que o trabalho era realizado por trabalhadores especializados. A revolução industrial foi um passo muito importante para a evolução dos sistemas de produção e por consequente introdução de novos tipos de produção, começando por ajudar a produção artesanal até à produção de produtos que muito dificilmente seriam possíveis artesanalmente.

Mesmo no mundo automóvel, em 1896, Ford construiu artesanalmente o seu quadriciclo, assim chamado devido aos seus pneus estreitos, parecidos aos de uma bicicleta. Nesta época a maioria dos carros eram construídos em quantidades limitadas, seguindo um tipo de produção artesanal. Essa seria uma das razões para que a empresa de automóveis de Ford não tivesse êxito. Todavia, mais tarde, reentra no negócio e, em 1908, a Ford lança no mercado dos Estados Unidos, o modelo T. Com base nas teorias de Taylor, o conceito de produção em massa, caraterizada pela produção do mesmo produto em larga escala, utilizando uma linha de montagem rígida, mudando o paradigma da existência de pequenas oficinas com pessoal altamente especializado para, grandes áreas de trabalho, com equipamento especializado e de elevado custo [7].

A introdução da linha de montagem de produção, em média o ciclo de trabalho de montagem, ou seja, o tempo que o operador trabalha, antes de repetir a mesma tarefa, foi reduzida de 514 minutos para 2,3 minutos [9].

O modelo de produção em massa requer estabilidade e controlo das variáveis de entrada, dos mercados e da força de trabalho. Mas a partir dos anos 70, estes parâmetros tornaram-se mais instáveis devido às flutuações económicas, ao aumento do poder do consumidor, à homogeneidade do mercado e com a globalização dos mercados.

A produção em massa idealizada por Ford foi incapaz de tratar as variações no tipo de produto. As empresas precisavam de se tornarem mais competitivas, a fim de cumprir as exigências do mercado para a redução de preços, melhor qualidade do produto, reduzir o tempo de entrega e diversificar a oferta. Por isso, a rigidez da produção em massa, passou a ser um obstáculo, para a produção de alguns produtos.

O *Just in Time* (JIT) foi uma filosofia de gestão de operações, introduzida por uma empresa japonesa, a Toyota Motor Inc, supervisionado pelo engenheiro-chefe Taichii Ohno, depois de ter estudado o modelo Ford [9]. Esta filosofia consiste em ter o material certo no lugar certo, na hora certa, eliminando desta forma, as existências e usando um controlo muito simples. Produzir em JIT requer um fluxo contínuo de materiais e de informação coordenados de acordo com um sistema *pull*, a trabalhar com um tempo de ciclo próximo ao do *takt time*, um tempo de ciclo que é definido em função da procura.

Muitas empresas, especialmente norte-americanas, estudaram o sistema de produção da Toyota, utilizando como base o conceito *lean production*, no qual o objetivo principal era eliminar o desperdício em todas as atividades, conseguindo fabricar produtos com menor tempo de projeto, menores existências, melhor qualidade e com reduzidos tempos de configuração.

Na década de 80, muitas empresas procuraram desenvolver paradigmas e tecnologias à procura da flexibilidade. Porém, nos anos 90, foram desafiados pela necessidade de aumentar a agilidade. A manufatura ágil, introduzida pelo *Instituto Iacocca na Universidade de Lehigh*, consiste na capacidade do sistema se adaptar de uma maneira rápida e rentável a mudanças contínuas e inesperadas no ambiente de produção [10].

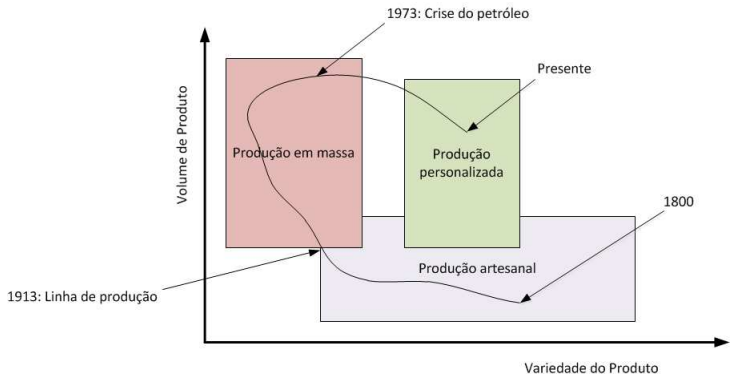

Figura 2.3: Evolução dos paradigmas de produção (Adapatado a partir de [7]).

O impacto da agilidade na manufatura, no projeto do produto, no relacionamento com clientes, bem como na produção tem sido expresso por quatro princípios subjacentes [11]: dar valor ao cliente, capacidade de reagir a alterações, o valor do conhecimento humano e as habilidades e capacidade de construir parcerias virtuais. Os três primeiros princípios podem ser encontrados no paradigma do *lean production*, o quarto princípio faz a diferença entre a *lean manufacturing* da manufatura ágil: na manufatura ágil as empresas formam alianças temporárias com outras empresas, até mesmo concorrentes, para reagir a situações inesperadas, com benefícios mútuos para ambas as empresas [10].

Nos dias de hoje, os conceitos de *lean manufacturing* e de manufatura ágil podem ser benéficos para as empresas que comecem do zero ou para empresas incapazes de suportarem grandes riscos financeiros. Ambos se baseiam no princípio de fornecer produtos de alta qualidade e de baixo custo com o mínimo de desperdício.

Com o constante aumento da procura de produtos cada vez mais personalizados a baixos preços no mercado, surgiu o conceitos *mass customisation* [12]. A *mass customisation* requer bons níveis de flexibilidade e agilidade, utilizando processos flexíveis, a partir das tecnologias de automação e estruturas organizacionais flexíveis para produzir

diversos produtos personalizados.

Este último paradigma mencionado pode ser uma boa abordagem para os dias de hoje, visto que as empresas operam cada vez mais em ambientes mais exigentes e imprevisíveis. E estas novas abordagens podem não ser sinónimo de obter produtos mais caros aos obtidos, por exemplo, pela produção em massa.

2.2 Classificação dos sistemas de produção

Os sistemas de produção podem ser classificados de acordo com o *layout* de produção e volume de produção [7].

A classificação realizada de acordo com o *layout* de um sistema de produção, corresponde à classificação do sistema de produção de acordo com a disposição das instalações físicas. Nas indústrias de produção, de itens discretos, é possível encontrar três *layouts* de produção principais: *fixed position*, *product flow layout* e *process layout*.

No esquema da *fixed position*, o produto não muda de posição, devido ao seu tamanho e peso, por isso são os operadores e as máquinas a se adaptarem ao produto.

No esquema do *product flow layout*, o equipamento está disposto numa linha de produção, de maneira a minimizar o tempo de transporte entre máquinas. O transporte pode ser realizado manualmente, através de transportadores automáticos, robôs e por *Automatic Guided Vehicle* (AGV). Este *layout* apresenta baixos tempos de transporte e um simples sistema de planeamento e controlo de produção. Mas apresenta pouca flexibilidade na mudança do produto e requer um investimento elevado, ao duplicar o equipamento na linha de produção.

No esquema de *process layout*, as máquinas são agrupadas de acordo com o processo de fabrico. Desta forma cada produto pode ter a sua sequência de operações. Este *layout* é adequado para o *batch production type*, pois apresenta flexibilidade e baixo investimento. No entanto, apresenta baixa eficiência no transporte de material e maior complexidade no sistema de planeamento e controlo de produção.

Os sistemas de produção classificados de acordo com o volume de produção podem ser divididos em três vectores [6]: *job shop*, produção em lotes e produção em massa.

No caso do *job shop*, são produzidas pequenas quantidades e geralmente com grande variedade de produtos. Por isso, o material utilizado para este tipo de produção deve ser flexível e deve suportar grande variedade de produtos. A produção de máquinas-ferramentas, moldes e aviões são alguns dos exemplos deste tipo de produção.

A produção em lotes corresponde à produção de quantidades médias do mesmo produto, na qual têm uma procura regular. Geralmente o equipamento utilizado neste tipo de produção é concebido para ter taxas de produção elevadas. É possível encontrar este tipo de produção na fabricação de equipamentos eletrónicos e de móveis.

A produção em massa está relacionada com a produção especializada a um pequeno número de produtos, com velocidades de produção elevadas. Como o objetivo passa por ter elevadas velocidades de produção, o equipamento e a planta da fábrica são dedicados à produção de um determinado produto. Este tipo de produção é utilizada, por exemplo na fabricação de parafusos e lâmpadas.

2.3 *Computer Integrated Manufacturing*

O paradigma CIM consiste na integração das atividades da empresa, relacionadas com a produção, de tecnologias de informação, como base de dados, que permita a troca e partilha de dados [13].

Para uma empresa conseguir responder aos desafios do mercado, tem que adotar meios que permitam, não só tratar eficientemente toda a informação necessária à sua operação, como também disponibilizar essa informação. Por exemplo, quando uma empresa apresenta uma base de dados para cada departamento, torna-se complicada a transferência de dados. A fim de eliminar o tempo de transferência de dados, este conceito propõe uma base de dados comum. Desta forma, os diferentes sectores, como o de marketing, *design* e planeamento de sistemas, vão se aproximar.

É fundamental interligar todos os níveis de uma empresa, de modo a tomar as melhores medidas, para que se possam aumentar a produtividade e os lucros. Por isso, é cada vez mais importante o modo como o fluxo de informação é planeado e implementado. O desafio de produzir de forma ágil, com grande flexibilidade, baixo custo e alta qualidade, exige que toda a informação envolva todos os intervenientes que necessitem, de forma simples e imediata, em tempo-real [14].

Como é ilustrado na figura 2.4, o modelo Y representa a integração de várias atividades organizativas e técnicas, ligadas e coordenadas, num ambiente de produção, através de um sistema de informação totalmente integrado. As atividades organizativas correspondem ao planeamento, gestão e controlo de produção. As atividades técnicas referem-se ao *Computer Aided Design* (CAD), *Computer Aided Manufacturing* (CAM), *Computer Aided Planning* (CAP), *Computer Aided Engineer* (CAE) e *Computer Aided Quality* (CAQ) [15].

No lado esquerdo do modelo estão as atividades de planeamento e controlo da produção, enquanto do lado direito estão as atividades técnicas de engenharia e produção. Na parte superior do modelo estão situadas as atividades relativas ao planeamento e na parte inferior a secção da implementação, onde são incluídas as atividades responsáveis pelo controlo da produção. No meio do modelo Y são representadas a base de dados que alimenta o fluxo de informações do sistemas, por exemplo com, dados sobre os equipamento, níveis de existências e listas de materiais [15].

As atividades operacionais estão relacionadas com o *Production Planning and Control* (PPC). Este corresponde às atividades desde que é realizada uma encomenda, passando pela estimativa de custos e pelo planeamento de produção, com ajustamento da capacidade, se for necessário, até à gestão de materiais, lançamento das ordens de compra e de produção. O estado das ordens de produção é controlado a partir da recolha de dados da planta fabril. Por sua vez, estes dados permitem controlar quantidades de fabrico, custo e tempos de operação.

Production Planning and Control

Para enfrentar as crescentes exigências dos consumidores, o PPC é um elemento decisivo na estratégia das empresas para enfrentar as crescentes exigências pela melhor qualidade, maior diversidade de produtos e produtos finais mais confiáveis. É um elemento central na estrutura de um sistema de manufatura, fundamental para garantir a eficiência e a eficácia do sistema produtivo [16].

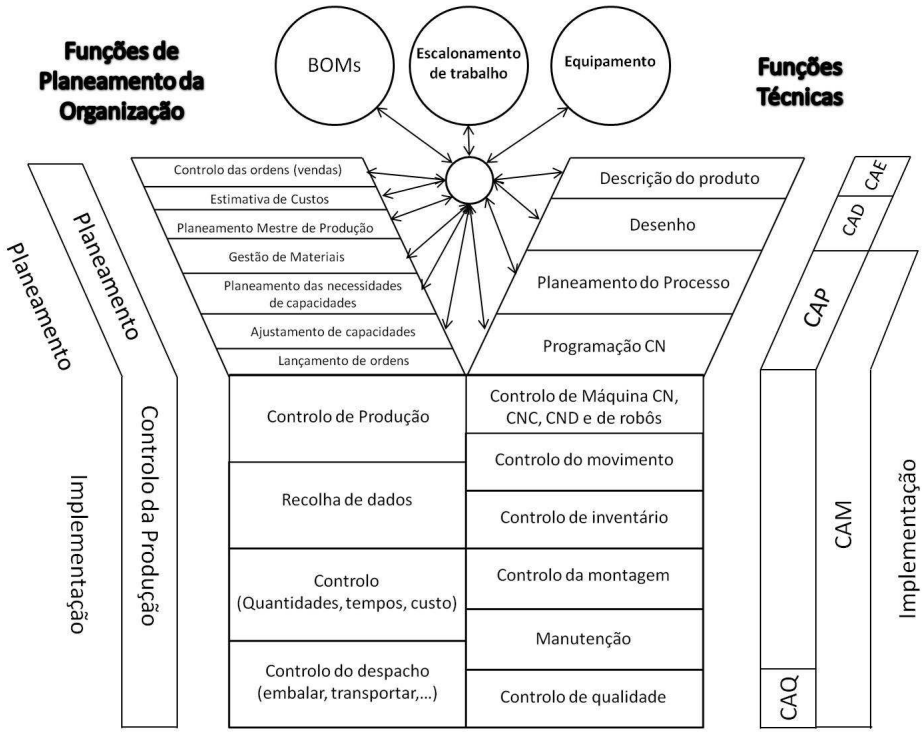

Figura 2.4: Modelo Y - Scheer do sistema (Adaptado a partir de [15]).

O PPC é baseado num conjunto de funções inter-relacionadas, para comandar o processo produtivo e coordená-lo com os vários sectores administrativos da empresa. A sua integração depende do tipo de indústria, dimensão e, as várias estruturas na empresa em questão [17].

A prioridade é que o PPC seja um sistema de apoio à produção, que comande e coordene o processo produtivo, fundamental para cumprir o planeamento e a programação dos processos de maneira eficaz, para assim satisfazer os requisitos de tempo, qualidade e quantidades do sistema produtivo.

Computer Aided Planning

O planeamento é um processo importante como elo de ligação entre o projeto e produção. É muito importante o plano de trabalhos desde o estado inicial até ao estado final do produto. A ferramenta CAP vêm por isso auxiliar à elaboração da sequência de operações, como o processamento, montagem e inspeção, necessárias para a produção do produto. É também de referir que esta fase tem como base as especificações feitas no desenho técnico, porque características como as propriedades dos materiais e tolerâncias podem ser influentes neste plano.

Os principais passos da elaboração do plano do processo são:

- seleção da matéria-prima;
- determinação da sequência de operações;
- seleção do tipo de máquinas que executam as operações;
- seleção das ferramentas;

- acessórios e equipamentos de inspeção;
- determinação de parâmetros de produção (velocidade de corte, avanço, etc.);
- determinação de tempos de fabricação (tempo para a preparação, tempo de processamento, tempo de maquinagem).

Computer Aided Design

O processo de desenho pode ser dividido nas seguintes fases [15]:

- Conceção;
- Desenvolvimento;
- Detalhe.

A fase de conceção corresponde à análise e avaliação de várias soluções e propostas. Na fase de desenvolvimento as soluções são avaliadas, por exemplo, através da construção de protótipos. Por fim, a fase de detalhe corresponde à representação e especificação das peças. Nesta última fase já é realizada a preparação das instruções para a produção.

Através da ferramenta CAD todo este processo de desenho pode sair beneficiado. O CAD tem como base os editores gráficos, constituídos por um conjunto de rotinas que, permitem a criação e manipulação de imagens compostas com o auxílio do computador. A sua utilização permite criar, atualizar e documentar um projeto. O tempo de desenvolvimento diminui e a qualidade é melhorada. Por isso, é evidente que esta ferramenta visa o aumento da produtividade de conceção do projeto.

Computer Aided Engineering

O CAE é uma ferramenta que complementa o CAD, pois baseia-se na construção e teste de protótipos, a nível de *software* e, permite avaliar a exequibilidade e funcionalidade do produto, uma vez que fornece ferramentas computacionais de análise mecânica, cinemática e de elementos finitos.

A análise realizada através desta ferramenta aprimora a qualidade do produto, visto que verifica se o produto obedece às características mecânicas e estruturais exigidas.

Computer Aided Manufacturing

A componente CAM refere-se ao controlo informatizado de transporte, armazenamento e das máquinas de produção. Tanto a produção autónoma como as formas organizacionais para a produção flexíveis podem ser abordadas através desta atividade.

As máquinas de controlo numérico foram um dos primeiros passos dados na produção assistida por computador. As máquinas de controlo numérico surgiram, nos anos cinquenta, depois de uma demonstração bem sucedida no Instituto de Tecnologia de *Massachusetts*. O uso das máquinas de controlo numérico permitiu melhorar substancialmente a qualidade e repetibilidade dos produtos produzidos. Por isso, nos anos 70, surgem as máquinas-ferramentas com Controlo Numérico Computorizado (CNC). As máquinas CNC equipadas com sistemas automáticos de alimentação de peças, armazém de ferramentas e troca de ferramentas automática, capazes de executar interpolações e compensações das dimensões das ferramentas, permitiram diminuir os tempos não produtivos. Neste campo surgiram também os sistemas CAD/CAM, que permitiram criar

automaticamente programas de maquinagem, pois a elaboração manual de programas é uma tarefa que pode dar muito trabalho e ser suscetível a erros humanos. A criação automática dos programas, a partir de o pós-processador configurado para a máquina em questão, veio reduzir o tempo de concepção de um programa, mas também a possibilidade de analisar melhor as estratégias de maquinagem utilizadas.

A introdução dos robôs permitiu o aumento da produtividade, da robustez, da velocidade e da resistência a ambientes adversos. Há muitas definições para os robôs, como a definição realizada pelo Instituto Americano de Robótica, que define *"o robô é um manipulador multi-funcional, programável, projetado para mover materiais, componentes, ferramentas ou dispositivos especiais através de movimentos programáveis variáveis para a execução de uma variedade de tarefas"*. Na indústria, em geral, é aplicado na pitura, soldadura, montagem ou em operações de manuseamento de material.

Os sistemas de produção automatizados podem ser otimizados se o fornecimento de ferramentas, peças e materiais de produção também for automático. O AGV, ilustrado pela figura 2.5, é um sistema de transporte de material inteligente, flexível e versátil. Alimentado a partir de baterias e, controlado por microprocessador, consegue-se mover numa trajetória pré-definida. É capaz de se mover em todas as direções, para carregar ou descarregar automaticamente materiais, mesmo em ambientes hostis.

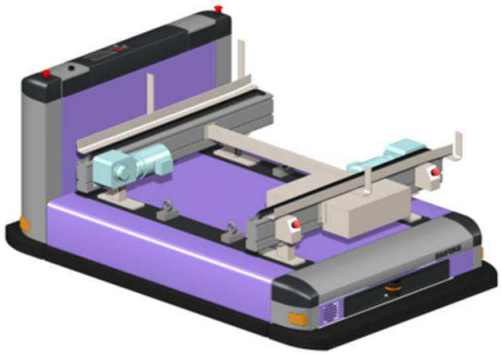

Figura 2.5: Sistema de transporte de material AGV [18].

Um *Automated Storage/Retrieval System* (AS/RS) é vocacionado para armazenar temporariamente materiais, através de dispositivos automáticos e gerido por tecnologias de informação, sem intervenção humana. Estes sistemas são amplamente utilizados, para armazenar matérias-primas, peças intermediárias, produtos acabados, ferramentas, peças para recuperar e peças não conformes. Apresenta-se como uma boa solução de armazenamento, visto que, otimiza o espaço de armazenamento, porque pode ter uma estrutura com elevada altura e com corredores estreitos. Também a nível superior, o sistema de controlo de produção é beneficiado, pois fornece informações, como as existências e datas de fornecimento no sistema, evita ruturas e possibilita uma gestão em tempo-real (ver figura 2.6).

Um centro de processamento é definido por uma máquina com controlo numérico, com troca de ferramenta automática e que pode lidar com a execução de várias operações de trabalho num processo contínuo. O processo de maquinagem pode ser considerado como um exemplo clássico de um centro de processamento. Os centros de processamento são utilizados em pequena a média dimensão de fabrico em série, e na produção de peças

muito complexas, mesmo em níveis de produção de pequeno porte. Integração de várias operações permite que o tempo de processamento seja bem reduzido [15].

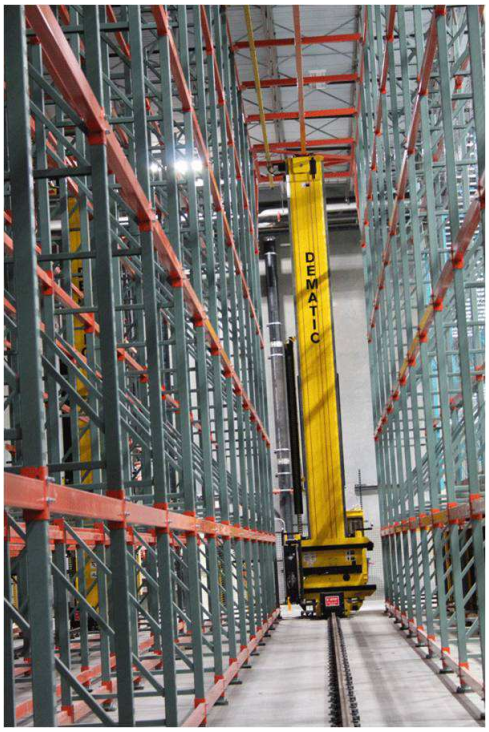

Figura 2.6: Sistema de armazenamento e recuperação automática [19].

Uma célula de fabrico flexível consiste em máquinas automatizadas, com um sistema de armazenamento intermédio e uma estação de carregamento e de aperto automático. Pode conter funções computorizadas a fim de proporcionar o controlo da ferramenta, variáveis especiais e monitoramento em tempo-real. Assim, uma célula de produção flexível é composta por várias máquinas-ferramentas controladas numericamente, que pode processar automaticamente partes de produção similares durante um período prolongado. Se a aquisição e a deposição de peças de produção também for automatizado, as células de produção, em seguida, são flexíveis [15].

O sistema de produção flexível consiste num sistema de processamento, em que o sistema de fluxo de materiais e o sistema de fluxo de informação, estão interligados entre si. Através de programas de controlo, o computador assume o transporte de peças de produção e das ferramentas, bem como o fornecimento às instalações de produção. A sequência de operações pode ser flexível, uma vez que o transporte não se baseia numa ordem específica de pistas da máquina. Uma vez que as estações de processamento são fornecidas com programas de controlo numérico, a partir do computador de controlo do sistema de produção flexível, isto pode ser interpretado como um sistema *Direct Numerical Control* (DNC). As estações de processamento individuais são em geral sistemas de CNC, mas também podem ser centros de processamento mais extensos [15].

Computer Aided Quality Assurance

Cada vez mais os sistemas de verificação das empresas de produção são automatizados, por exemplo, com sensores e sistemas de visão, e o planeamento do processo de verificação apoiado por sistemas informáticos. Os procedimentos de teste podem ser apresentados através da estatística e pesquisas para planear operações de controlo de qualidade.

As questões de qualidade e controlo acompanham o processo desde a verificação dos materiais de entrada, o controlo de qualidade do processo de produção e, o controlo do produto final. Segundo [15], a descoberta tardia de qualquer erro pode levar a grandes custos, ou por outras palavras, a garantia da qualidade pode constituir até 50% dos custos de produção.

Vantagens

As principais vantagens do paradigma CIM podem ser listadas como [13; 20]:

- Aumento da produtividade: a eliminação da redundância de informação conduz a uma melhor gestão e controlo dos recursos, com melhorias a rondar os 40 a 70%;
- Aumento da flexibilidade: uma vez que a informação é partilhada é possível ter um controlo descentralizado, o que leva a melhorar a rapidez de resposta a perturbações externas e internas;
- Aumento da qualidade: a integração de sistemas automatizados reduz o número de falhas, uma vez que a informação nunca é duplicada. Com o sistema CIM é possível aumentar 2 a 5 vezes a qualidade do produto;
- Redução do tempo do projeto: uma vez que a informação do desenho do produto é partilhada por todas as equipas responsáveis, consegue-se obter uma redução na ordem dos 15 a 30% no tempo de desenho no projeto;
- *Work in Process* (WIP): este tipo de gestão integrada permite uma redução de 30 a 60% do trabalho em progresso.

2.4 Arquiteturas das Comunicações Industriais

O CIM engloba todas as atividades nos sistemas de fabrico, não só as operações de fabrico, como a coordenação e cooperação entre os diferentes subsistemas. As comunicações são um aspeto importante, uma vez que as atividades de fabrico desenvolvidas, em ambiente CIM, traduzem-se essencialmente pela transferência, armazenamento e processamento de informação. Outro fator a ter em conta passa pela heterogeneidade do ambiente fabril, no que toca aos equipamentos, por isso é importante definir normas nos sistemas de comunicação.

As comunicações requerem infraestrutura técnicas, como o *software* e o *hardware*, e regras que condicionem aspectos técnicos e funcionais, como os protocolos. É fundamental que estas tenham capacidade de tornar a informação de uma localização geográfica acessível para outra e, por outro lado, com capacidade de transferir informação de um ponto para o outro com eventual necessidade de armazenamento de informação por períodos definidos ou indefinidos de tempo.

Segundo a *International Organization for Standardization* (ISO), os sistemas de fabrico automatizados podem ser hierarquizados entre quatro a seis níveis. Mesmo que

seja referente às indústrias de fabrico discreto, o modelo, ilustrados na figura 2.7, pode também ser aplicado ao controlo de processos [3].

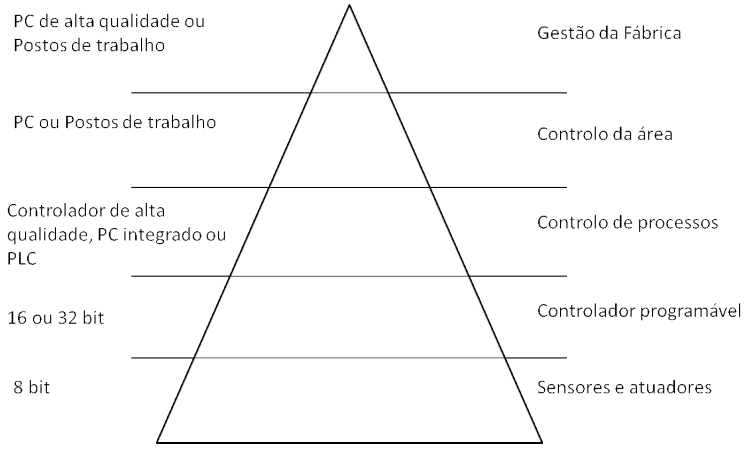

Figura 2.7: Modelo hierárquico dos sistemas de fabrico automatizados.

A divisão em distintos níveis é baseada, entre outros aspectos, nos tipos de atividades realizadas nas empresas, e geralmente, na utilização de diferentes tipos de redes de comunicação nos diferentes níveis. A partir deste modelo os fluxos de informação podem ser mais percetíveis. Os fluxos verticais correspondem aos fluxos entre entidades de níveis hierárquicos adjacentes e os fluxos de informação horizontais aos fluxos entre entidades do mesmo nível.

Nas áreas do topo da pirâmide, relacionadas com o planeamento ou com a engenharia de conceção, as comunicações devem satisfazer trocas com grandes quantidades de informação, nas quais têm de ser processadas durante períodos relativamente baixos.

Os níveis inferiores da hierarquia geralmente trocam pequenas quantidades de informação, que necessitam ser processadas de forma rápida, porque a baixo nível são controlados processos industriais de tempo crítico. Este tipo de transações tem normalmente uma periodicidade cíclica e frequência relativamente elevada.

As redes para os níveis mais baixos da hierarquia das redes de comunicação são designadas por redes de campo ou *fieldbus*, que são basicamente vocacionadas para interligar sensores, atuadores e controladores.

As atividades relacionadas com o controlo do processo industrial estão intimamente ligadas à estrutura de comunicações que lhes serve de suporte. Daí surge também a adoção de um modelo hierárquico para a arquitetura de comunicações. A estrutura de controlo pode variar em número de níveis, entre quatro e seis, mas ao nível da arquitetura de comunicações é usual estar dividido em três.

Dificilmente seria capaz de existir uma única rede de comunicação capaz de satisfazer os requisitos necessários para as trocas de informação existentes num ambiente industrial, devido às distintas características que o ambiente pode ter. Nos níveis inferiores da hierarquia CIM, podem ser abordadas diferentes soluções.

2.4.1 Redes de Fábrica

As redes de fábrica são vocacionadas para os níveis superiores da hierarquia da estrutura de controlo. A este nível são exercidas atividades de planeamento de produção, de

processo e de materiais e as áreas de engenharia financeira e comercial.

O fluxo de informações carateriza-se essencialmente para os níveis inferiores, nas ordens de fabrico e nas informações associadas ao seu escalonamento. E para os níveis superiores, relativas ao estado das ordens de fabrico, à qualidade do processo produtivo e aos pedidos de aquisição de materiais e recursos.

Por isso, as redes de fábrica distinguem-se pelo facto de terem de apresentar características que permitam um elevado fluxo de informação sem requisitos temporais críticos.

2.4.2 Redes de Célula

As redes de célula procuram satisfazer as necessidades intermédias da hierarquia. Como foi mencionada anteriormente, uma célula agrupa um conjunto de equipamentos que cooperam para a execução de uma tarefa.

A este nível pretende-se corresponder a atividades como o escalonamento, o sequenciamento, execução de tarefas, recolha dos dados da qualidade de produção e recolha dos dados relativos ao desempenho dos equipamentos constituintes de uma célula.

Estas redes são pretendidas para transferir para níveis descendentes informação que podem conter, por exemplo, ordens de execução de operações ou programas de controlo, e em sentido ascendente devem disponibilizar informação sobre a evolução e resultados das operações executadas.

Por isso, este nível apresenta fluxos de informação de volume intermédio com requisitos de tempo exigentes, mas que podem não ser críticos.

2.4.3 Redes de Campo

As redes de campo apresentam características vocacionadas para o controlo direto do processo industrial, nomeadamente a execução de algoritmos de controlo, nos equipamentos ou dispositivos que atuam fisicamente no processo dos produtos a operar. Desta forma, a interface com o processo é realizada através de sensores e atuadores, em que alguns podem apresentar capacidades de processamento complexas.

A este nível os fluxos de informação apresentam, por isso, um reduzido volume, mas com requisitos temporais críticos.

O modelo de referência *Open Systems Interconnection* (OSI) serve de base para descrever a arquitetura conceptual de um sistema de comunicação genérico. Mas em aplicações de tempo-real os recursos como a capacidade de processamento e a quantidade de memória por vezes são limitados. Como é desejável também a menor latência possível, o modelo de referência OSI muitas vezes não é totalmente implementado, podendo designar a esta implementação parcial, representada pela figura 2.8, de *Enhaced Performance Architecture* EPA [21; 22].

A camada física é responsável pela codificação, descodificação e temporização dos bits, pela sincronização e pelas características físicas do barramento, como a topologia da rede, o canal físico utilizado para a comunicação, a taxa de transmissão, o tamanho máximo do barramento, o número máximo de nós que podem ser ligadas à rede e a imunidade às interferências eletromagnéticas [23].

A camada da ligação de dados ocupa-se da transferência de informação entre vários pontos da rede, encapsulando/desencapsulando a informação sob a forma de tramas, pelo controlo do acesso ao meio e pela deteção, sinalização e limitação de erros [24].

A camada de aplicação seleciona os serviços com as funções apropriadas para cada uma das aplicações [24].

Existem diferentes tipos de topologias possíveis para as infra-estruturas de comunicação de tempo-real em sistemas distribuídos: barramento, estrela, anel, malha ou árvore, contudo as mais utilizadas em sistemas de tempo-real são as topologias em barramento e em anel [25; 26].

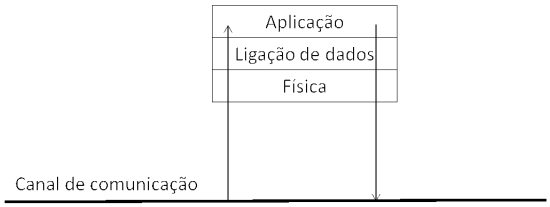

Figura 2.8: Camadas do modelo EPA (Adaptado a partir de [24]).

2.5 Arquitetura dos Sistemas de Controlo

A crescente descentralização ao nível das funções de controlo e a crescente utilização de dispositivos inteligentes baseados em microprocessadores ou microcontroladores, criaram condições necessárias para o desenvolvimento das redes de campo.

Um sistema de controlo tem um papel relevante para o funcionamento correto e seguro de um processo industrial de qualquer natureza. Independentemente da complexidade e dimensão do processo industrial em causa, este pode ser decomposto em três subsistemas, como é representado pela figura 2.9: processo controlado, controlador e operador humano.

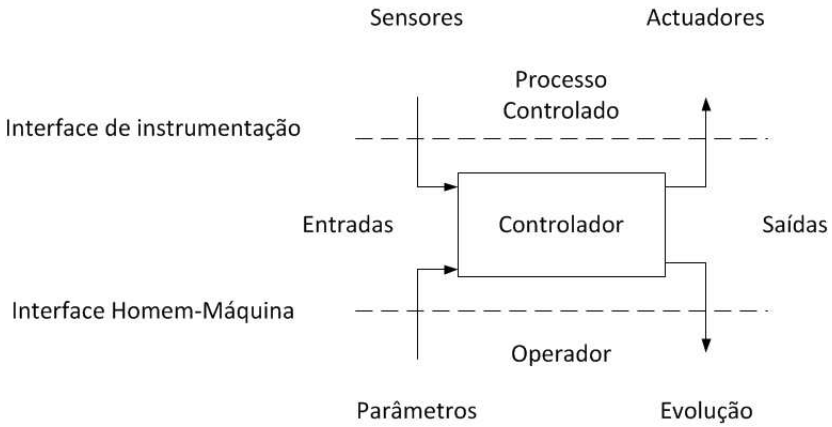

Figura 2.9: Sistema de controlo.

O controlador é um equipamento intermediário, uma vez que interage através de duas interfaces distintas com os outros dois subsistemas.

A interface que interage com o processo controlado, definida como a interface de instrumentação, consiste num conjunto de sensores e atuadores que transformam os sinais físicos do processo controlado em sinais com características apropriadas para serem utilizados pelo controlador, e vice-versa. Um sensor é um dispositivo eletrónico sensível a

uma quantidade física e capaz de a converter num sinal mensurável por um utilizador. Os sensores podem ser de vários tipos, desde sensores de temperatura, humidade, luminosidade, pressão, entre outros. Os sensores são por isso um componente importante na malha de controlo, visto que permite que o controlador conheça o estado do sistema. Na sua ausência qualquer tipo de controlo por realimentação seria impossível ser realizado. O atuador é um elemento da malha de controlo que pretende controlar o sistema. A sua localização física é muito importante, tendo muitas vezes de seguir requisitos muito específicos, através de mecanismos mecânicos, pneumático ou elétricos.

A interface que interage com o operador humano, definida como a interface homem-máquina ou *Human Machine Interface* (HMI), consiste num conjunto de dispositivos de entrada e saída, que permitem a interação com um operador humano. Esta interação tipicamente realiza-se ao nível da definição dos parâmetros do processo e da supervisão da respetiva evolução. Muitas vezes deve ser colocada num local remoto do sistema a controlar, como, por exemplo, em fábricas onde são muitas vezes centralizadas em centros de comando, para uma gestão global simplificada.

Basicamente, o controlador é responsável por controlar a evolução do processo através da execução de um algoritmo de controlo adequado, a partir do processamento da informação oriunda da interface de instrumentação e da HMI. Para realizar estas funções o controlador dispõe de uma estrutura funcional, baseada na utilização de equipamentos adequados ao processo em causa, que suporta a execução do algoritmo de controlo.

2.5.1 Arquitetura Centralizada

A arquitetura comum nos sistemas de automação industrial e de controlo era a centralizada, até aos anos 80. Como é apresentada pela figura 2.10, a arquitetura centralizada consiste num ponto central onde todo o processamento de controlo é realizado e onde são idealizados todos os comandos para serem aplicados ao processo [23].

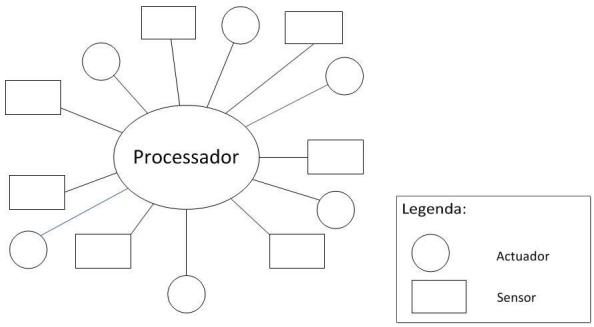

Figura 2.10: Sistema de controlo centralizado.

O tipo de topologia em estrela acarreta alguma simplicidade, porque a informação está centralizada e é sincronamente coordenada por apenas uma unidade processadora. Mas algumas inconveniências podem ser levantadas, uma vez que os sensores e/ou atuadores podem estar dispersos, originando alguns problemas como [23]:

- Uma grande concentração de cablagem junto do controlador, com os problemas de atravancamento inerentes, resultando em dificuldade de instalação e manutenção;
- Uma extensão potencialmente muito grande da cablagem total, com um elevado custo associado;

- Sensibilidade ao ruído, muito em parte porque a maioria das transmissões I/O eram analógicas.

Este tipo de arquitetura apresenta também baixa fiabilidade, uma avaria da unidade central comprometeria o funcionamento de todo o conjunto. Este inconveniente pode ser crítico, se pela natureza ou condicionante económica não seja aceitável que possa ocorrer uma interrupção do processo. O número de entradas e saídas estão limitadas, tendo em conta, que um possível aumento do sistema possa originar mais processamento, de tal forma que a unidade central possa não ter recursos suficientes para processar toda a informação [27].

2.5.2 Arquitetura Distribuída

Problemas como alguns mencionados na arquitetura centralizada impulsionaram outras soluções, como os sistemas de controlo distribuído. Várias funções de controlo, como o cálculo e a supervisão podem estar distribuídos pelos vários elementos do sistema, normalmente designados nós ou nodos.

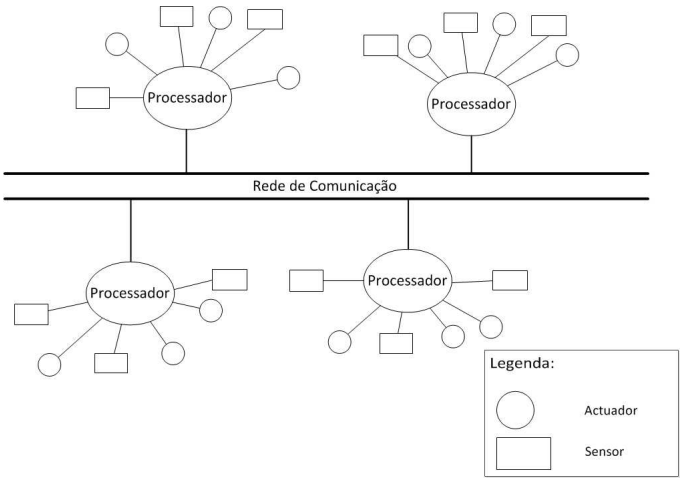

Figura 2.11: Sistema de controlo distribuído.

A figura 2.11, clarifica que cada nó tem autonomia de processamento e cada um destes elementos faz interface com o processo através dos respetivos componentes de I/O e com os restantes nós do sistema através de um sistema de comunicação. Apenas é necessário um canal de comunicação comum a todos para interligá-los, mas desta forma é necessário que haja um sistema de comunicação que suporte todas as trocas de informação entre os vários nós e que garanta o bom funcionamento do conjunto, com níveis de segurança aceitáveis para que a informação não seja adulterada e chegue íntegra ao seu destino.

O algoritmo de controlo já não se executa apenas numa única unidade controladora, onde é relativamente fácil sincronizar todas as atividades desempenhadas. Passa a ser composto por múltiplos programas que requerem sincronização e comunicação adequadas.

Apesar desta arquitetura apresentar simplificação da cablagem, a complexidade da sua implementação é maior, contudo apresenta outras vantagens, como a maior liberdade ao crescimento do sistema, ao nível da capacidade de processamento e da comunicação. Será possível acrescentar nós para completar necessidades de processamento, desde que

haja capacidade de comunicação disponível. Se a capacidade de comunicação ficar esgotada, a solução passa pela descentralização do processamento da informação, criando módulos com processamentos independentes, de maneira a que a informação circule pelas vias de comunicação, comuns a todos os módulos, para que a informação seja recebida por todos, e estes as utilizem em função das suas necessidades. Quando se criam módulos especializados num determinado tipo de processamento, cada um deles faz o seu processamento independente e ao mesmo tempo dos restantes módulos [1].

Genericamente quando se trata de sistemas de controlo distribuído poderão advir certas vantagens como [1]:

- Otimização de recursos: Numa aplicação distribuída é inerente um tipo de processamento paralelo, otimizando assim a gestão de tempos e recursos. Com a capacidade de cálculo de cada nó, a informação recolhida poderá ser tratada, ao linearizar e digitalizar a informação, para ficar menos susceptível a erros durante uma transmissão e diminuir a quantidade de informação transportada na rede;
- Modularidade: A capacidade de processamento dos nós possibilita que funcionalidades específicas sejam atribuídas a cada nó. Ao interligar os vários nós através de uma rede conduz a uma estruturação mais modular e, consequentemente, menos complexa, beneficiando desta forma na manutenção e na evolução destes sistemas;
- Cablagem: O uso de uma rede partilhada torna a cablagem mais simples e com mais vantagens. Por exemplo, num sistema baseado num barramento de campo, é possível fazer circular um número elevado de sinais diferentes sobre o mesmo meio geralmente com um cabo com dois ou quatro condutores;
- Escalabilidade: Esta característica indica a habilidade de manipular uma porção crescente de trabalho, de forma uniforme, ou estar preparado para adicionar novos elementos;
- Tolerância a falhas: Os sistemas de controlo distribuído conseguem replicar o mesmo programa em vários nós criando uma redundância espacial. Assim se uma das réplicas falhar outra poderá a substituir;
- Partilha de dados: Como a informação não está centralizada, a rede proporciona a facilidade na partilha de informação.

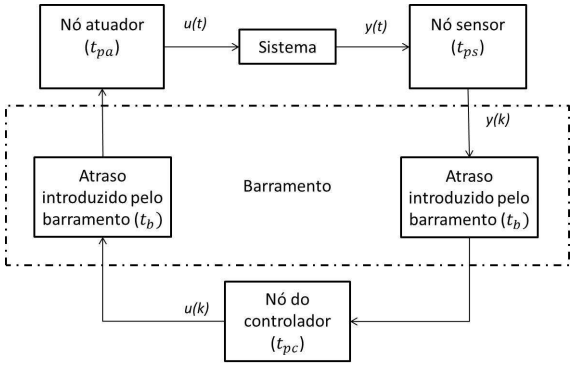

Figura 2.12: Diagrama de blocos detalhado de um sistema de controlo distribuído (Adaptado a partir de [24]).

A figura 2.12, apresenta o diagrama de blocos detalhado de um sistema de controlo distribuído, assim como os tempos de processamento e de comunicação associados ao seu funcionamento. O diagrama clarifica a proveniência de possíveis atrasos que podem estar associados tanto ao processamento das tarefas em cada nó como à utilização do barramento. É possível identificar os atrasos que medeiam entre o instante de amostragem e o instante de atuação. Esses atrasos são de dois tipos: tempo de processamento interno em cada um dos nós, como t_{ps}, t_{pc}, t_{pa}, e o tempo introduzido pelo mecanismo de controlo do acesso ao barramento, como t_b. Esses atrasos levam a que exista incerteza quanto ao instante de atuação sobre o sistema. Pode também existir incerteza em relação ao instante de amostragem, dependendo do modo como é feita a ativação da tarefa responsável pela mesma.

O tempo de acesso ao barramento depende do método de controlo de acesso ao meio específico de cada barramento, do tipo de tráfego e dos mecanismos de escalonamento utilizados [24; 28].

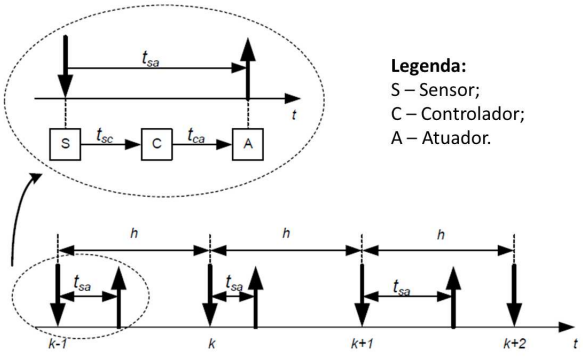

Figura 2.13: Representação temporal do tempo entre a amostragem e a atuação (Adaptado a partir de [24]).

A figura 2.13, apresenta os tempos envolvidos entre o instante de amostragem e o instante de atuação para cada ciclo de controlo. O período de amostragem é representado por h. As setas para cima e para baixo representam, respetivamente, o instante de amostragem e o instante de atuação. Através dessa imagem facilmente se pode deduzir o tempo entre a amostragem e a atuação, sabendo que este é variável, de ciclo para ciclo de controlo, e originando por isso incerteza no instante de atuação:

$$t_{sa} = t_{sc} + t_{ca} \tag{2.1}$$

Essa incerteza é designada, neste contexto, por *jitter*. Podem-se obter outras expressões mais pormenorizadas para t_{sa}. Basta comparar a figura 2.13 com a figura 2.12 para se obterem as expressões:

$$t_{sc} = t_{ps} + t_b \tag{2.2}$$

$$t_{ca} = t_{pc} + t_b + t_{pa} \tag{2.3}$$

O valor de t_{sa} varia consoante o tráfego presente no barramento e o mecanismo de escalonamento utilizado [24].

2.5.3 Arquitetura Centralizada e Distribuída

Em termos conceptuais, os aspectos que mais contrastam a arquitetura distribuída com a centralizada são:

- A aplicação de sistema de comunicação partilhado;
- O barramento de campo, que permite introduzir uma grande simplificação de cablagem e redução do respetivo custo;
- A distribuição do algoritmo de controlo, o qual pode ser dividido em várias partes que executam paralelamente em cada um dos nós;

A informação recolhida deve ser a mais completa, correta e atual sobre o estado do processo, para que o controlo seja realizado de forma mais eficaz quer em termos da qualidade do controlo.

A integração e aplicação em larga escala requer uma arquitetura adequada que sustente as interações necessárias à concretização dos objetivos globais do sistema. A arquitetura dos sistemas de automação industrial e de controlo tem vindo a evoluir ao longo dos tempos, havendo hoje um claro domínio das arquiteturas distribuídas, as quais apresentam inúmeros benefícios relativamente às arquiteturas mais antigas, baseadas num modelo centralizado de controlo e operação.

2.6 Sistemas de Controlo Distribuído em Tempo-Real

Os avanços tecnológicos aliados com diminuição dos custos de implementação facilitaram a introdução dos sistemas de controlo de tempo-real distribuídos. Hoje em dia, este tipo de sistemas estão presentes em dispositivos que interagem com o quotidiano de uma forma direta, mas pelo o facto de serem incorporadas faz com que os utilizadores finais não tenham muitas vezes consciência da sua importância, ou mesmo da sua existência.

Um sistema designa-se por tempo-real quando a correção do seu comportamento depende não só do valor final resultante da sua ação mas também do instante temporal em que esse valor é produzido. Os sistemas de tempo-real podem classificar-se de acordo com o tipo de restrições temporais, em *hard* ou *soft* [29].

Um sistema de tempo-real é do tipo *hard* se a produção do resultado para além do instante temporal imposto pela *deadline* pode originar falhas catastróficas na operação do sistema ou no ambiente em que este se insere. Neste tipo de sistemas a falha produzida pode ter consequências fatais, como por exemplo, no caso de uma passagem de nível automática. Se a cancela não for fechada dentro de um período de tempo bem determinado após a deteção da aproximação do comboio pode dar-se um acidente que pode resultar na perda de vidas [24].

Se por outro lado o *deadline* puder ser ocasionalmente ultrapassado, com perda do desempenho do sistema mas sem consequências desastrosas, então o sistema de tempo-real diz-se do tipo *soft*. Um exemplo típico, são as aplicações multimédia, como por exemplo, o *streaming* de vídeo [24].

Para que os sistemas de controlo estejam distribuídos, o meio de comunicação tem de apresentar elevados níveis de rendimento e fiabilidade necessários para este tipo de aplicações. Deve garantir o transporte da informação dentro de um intervalo de tempo limitado pelo *deadline*.

De acordo com [27], um sistema de comunicação de tempo real deve promover a modularidade, permitir a flexibilidade, proporcionar mecanismos de deteção de erro, apresentar um latência previsível, pequena e com um mínimo de *jitter* possível.

Comunicação *Time-Triggered* e comunicação *Event-Triggered*

A comunicação neste tipo de redes geralmente pode ter duas abordagens diferentes: a abordagem *Time-Triggered* (TT) e a abordagem *Event-Triggered* (ET).

Na abordagem TT a comunicação ocorre em instantes temporais pré-definidos. Uma das vantagens desta abordagem é o facto de permitir o estabelecimento de um desfasamento relativo entre as mensagens que são enviadas para o sistema de comunicação [30]. Esta abordagem é muito utilizada para o tratamento das mensagens periódicas.

Na abordagem ET a comunicação é iniciada pela ocorrência de um dado acontecimento, como a alteração de um valor de uma entrada digital. Esta abordagem é a mais adequada para o tratamento de mensagens aperiódicas.

Ambas as abordagens apresentam vantagens e desvantagens mas as conclusões são dispares, uma vez que estão dependentes das características que o sistema deverá ter.

Controlo de acesso ao meio

Neste campo algumas estratégias podem ser adotadas por causa da sua influência direta no atraso introduzido no sistema de comunicação. As estratégias ao nível do MAC podem se dividir em dois tipos: controladas e não controladas [31]. Nas estratégias controladas um sinal de controlo explícito ou implícito determina quando é que um determinado nó pode transmitir. Nas estratégias não controladas a arbitragem pelo acesso ao meio baseia-se no estado do barramento e em informação local. Cada um destes grupos ainda pode ser subdividido em dois: centralizado e distribuído. As estratégias mais comuns nos protocolos de controlo distribuído são:

- ***Carrier Sense Multiple Access with Collision Detection***: Esta passa por uma estratégia distribuída e não controlada. De acordo com esta estratégia um nó que deseje transmitir monitoriza o estado do barramento e começa a transmitir quando este não estiver a ser utilizado. Se acontecer uma colisão os nós envolvidos param de transmitir e voltam a tentar depois de um intervalo de tempo aleatório. Este tipo de estratégia é adotada pelo protocolo Ethernet (IEEE 802.3). Devido à dificuldade em determinar a latência máxima das mensagens este protocolo não foi inicialmente considerado apropriado para comunicações em tempo-real.

- ***Carrier Sense Multiple Access with Non-destructive Bitwise Arbitration***: Este MAC é também do tipo não controlado e distribuído. Esta estratégia permite que o nó transmite quando o barramento estiver livre, tendo também em consideração uma priorização para as mensagens. A arbitragem é feita bit a bit utilizando o valor do identificador e sem eliminar mensagens com menor prioridade em caso de colisão.

- ***Token-Ring***: A estratégia *Token-Ring* é do tipo controlado de forma centralizada ou distribuída. O direito de transmitir uma mensagem é concebido pela posse do designado testemunho. Na versão distribuída os nós o testemunho circula pela disposição em anel virtual. O nó que tiver a posse do testemunho transmite até se

esgotar um determinado tempo, ou no caso de isso não chegar a acontecer, até não ter mais mensagens para enviar.

- **Controlo centralizado num mestre**: Neste MAC o acesso dos nós ao barramento é controlado por um dispositivo com diferentes características dos outros, designado por mestre. Como é evidente este MAC é do tipo controlado e de forma centralizada. No modelo mestre/escravo o nó mestre concede explicitamente, através de uma mensagem, o direito de um escravo transmitir.
- ***Time Division Multiple Access***: Este MAC é do tipo controlado e distribuído. Nesta estratégia a concessão do acesso ao barramento é controlado pela passagem do tempo. Esta estratégia assenta numa base de tempo global e acessível em todos os nós. O tempo global é dividido em janelas. Durante cada janela temporal apenas um nó pode aceder ao barramento, que faz com que esta seja estática e previsível.

Capítulo 3

Redes de comunicação

Os protocolos *fieldbus* são redes industrias que surgiram pela necessidade de controlo em tempo-real e no controlo distribuído para interligar dispositivos e máquinas no chão de fábrica, criando um sistema que pode ser facilmente controlado e confiável. Os protocolos *fieldbus* podem ser baseados em padrões de protocolos de comunicação série, como o RS232 ou o RS485, como baseados pela tecnologia Ethernet. Na tabela 3.1 são apresentados alguns protocolos *fieldbus*:

Protocolos baseados em RS232 e RS485	**Protocolos baseados em Ethernet**
CANopen	EtherCAT
DeviceNet	Ethernet Powerlink
ControlNet	Ethernet/IP
PROFIBUS	PROFINET
Modbus-RTU	Modbus-TCP
LonWorks	Lon sobre Ethernet
SERCOS I/II	SERCOS III
CC-Link	CC-Link

Tabela 3.1: Alguns protocolos *fieldbus* utilizados no âmbito industrial [32].

3.1 DeviceNet

O protocolo DeviceNet é uma implementação do protocolo *Common Industrial Protocol* (CIP) para redes de comunicação industrial, que foi apresentado em 1994. Originalmente foi desenvolvido pela Allen-Bradley, mas mais tarde esta tecnologia viria a ser transferida para *Open DeviceNet Vendor Association* (ODVA). DeviceNet é um protocolo de alto-nível, uma vez que as duas camadas inferiores do modelo de referência OSI são implementadas pelo protocolo CAN [33].

O protocolo segue o modelo produtor/consumidor, suporta vários modos de comunicação e possui prioridade entre as mensagens, características que fazem com que este seja utilizado principalmente para a interligação de controladores industriais e dispositivos I/O. Pode ser configurado para operar numa arquitetura mestre/escravo ou numa arquitetura distribuída ponto-a-ponto. A rede DeviceNet pode conter no máximo 64 dispositivos e além disso possui mecanismos de deteção de endereços duplicados e isolamento

dos nós em caso de falhas críticas.

Utiliza uma topologia de rede do tipo tronco/derivação que permite que tanto os fios dos sinais como a alimentação estejam presentes no mesmo cabo. Ao nível da camada física do protocolo em questão, como é a mesma que o protocolo CAN devem ser instaladas resistências de terminação nas extremidades da rede.

O comprimento total da rede varia de acordo com a taxa de transmissão utilizada, conforme a tabela 3.2 apresenta.

Taxa de transmissão	Tamanho da rede	Derivação	
		Máximo	Total
125 kbps	500 m	6 m	156m
250 kbps	250 m		78m
500 kbps	100 m		39m

Tabela 3.2: Tabela com as taxas de transmissão e tamanho de rede possíveis no protocolo DeviceNet [33].

O protocolo DeviceNet possui dois tipos básicos de mensagens, I/O e *explicit* (ver figura 3.1). Cada um deles é adequado a um determinado tipo de dados, conforme descrito abaixo [33]:

- I/O: tipo de mensagem síncrono dedicado à movimentação de dados prioritários entre um produtor e um ou mais consumidores. Dividem-se de acordo como o método de troca de dados. Os principais são:
  - *Polled*: método de comunicação em que o mestre envia uma mensagem a cada um dos escravos da sua lista. Assim que recebe a solicitação, o escravo responde prontamente a solicitação do mestre. Este processo é repetido até que todos sejam consultados, reiniciando o ciclo.
  - *Bit-strobe*: método de comunicação onde o mestre envia para a rede uma mensagem com 8 bytes de dados. Cada bit destes 8 bytes representa um escravo que, se endereçado, responde de acordo com o programado.
  - *Change of State*: método de comunicação onde a troca de dados entre o mestre e escravo ocorre apenas quando houver mudanças nos valores controlados, até um certo limite de tempo. Quando este limite é atingido, a transmissão e receção ocorrerão mesmo que não tenha havido alterações. A configuração desta variável de tempo é feita no programa de configuração da rede.
  - *Cyclic*: outro tipo de comunicação muito semelhante ao anterior, a única diferença fica por conta da produção e consumo de mensagens. Neste tipo de comunicação, toda a troca de dados ocorre em intervalos regulares de tempo, independentemente de terem sido alterados ou não. Este período também é ajustado no *software* de configuração de rede.
  - *Explicit*: Tipo de mensagens de uso geral e não prioritário. Utilizado principalmente em tarefas assíncronas, tais como a parametrização e configuração do equipamento.

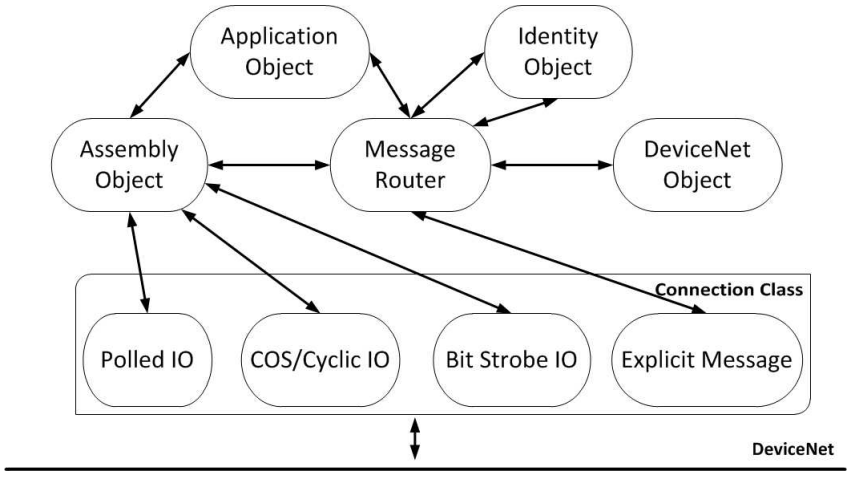

Figura 3.1: Esquema da comunicação DeviceNet.

3.2 PROFIBUS

O *PROcess Field BUS* (PROFIBUS) é um padrão de rede de comunicação industrial, utilizado num amplo espectro de aplicações, desenvolvido com o intuito de fornecer um modo de comunicação determinístico entre os diversos componentes de uma rede de controlo distribuído. O desenvolvimento do PROFIBUS foi iniciado em 1989, pelo Ministério Federal Alemão de Pesquisa e Tecnologia, em cooperação com alguns fabricantes de automação. Definido originalmente pela norna Alemã DIN 19245, hoje faz parte na norma internacional IEC 61158. Atualmente o PROFIBUS é baseado em padrões reconhecidos internacionalmente, sendo a sua arquitetura baseada no modelo de referência OSI conforme o padrão internacional ISO 7498 e apresenta também interdependência de fabricantes garantidas pelas normas EN50170 e EN50254 [34].

O PROFIBUS especifica as características técnicas e funcionais de um sistema de comunicação industrial, através do qual, dispositivos digitais podem se interconectar, desde o nível de campo até ao nível de células. É um sistema multi-mestre que permite a operação conjunta de diversos sistemas de automação e engenharia ou visualização.

A interação de dispositivos de diferentes fabricantes é garantida sem a necessidade de qualquer adaptação na interface. Usa conetores 9-pin D-type ou conetores de 12 mm *quickdisconnect*. O número de nós é limitado, aos 127, e a distância máxima suportada é de 24 km, recorrendo a *repeaters* e à transmissão por fibra ótica, com taxas de transmissão que podem variar entre 9600 bps até 12 Mbps. O comprimento do campo de dados de uma mensagem pode chegar aos 244 bytes. De acordo com a aplicação, pode-se utilizar como meio de transmissão qualquer um dos seguintes padrões: RS-485, IEC 61158-2 ou Fibra Ótica [34].

O PROFIBUS diferencia os seus dispositivos entre mestres e escravos. Os dispositivos mestres controlam o barramento, como a atribuição de acesso ao meio de transmissão. Um mestre, ou também designado por estação ativa, pode enviar mensagens sem uma requisição externa, sempre que possuir o direito de acesso ao barramento. Os dispositivos escravos, também designados por estações passivas, são tipicamente dispositivos periféricos. Eles não têm direito de acesso ao barramento e só podem enviar mensagens ao mestre ou reconhecer mensagens recebidas quando solicitados (ver figura 3.2).

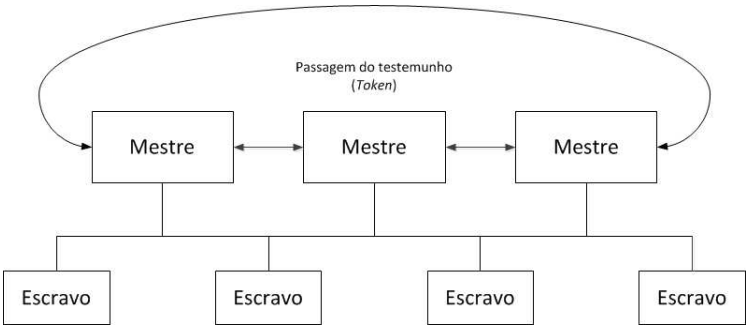

Figura 3.2: Esquema de funcionamento do protocolo PROFIBUS (Adaptado a partir [35]).

O PROFIBUS usa somente as duas camadas mais inferiores e bem como a última camada do modelo de referência OSI. Esta arquitetura simplifica e assegura uma transmissão de dados eficiente e rápida. O *Direct Data Link Mapper*(DDLM) proporciona à interface do utilizador um fácil acesso à camada de ligação de dados. As funções de aplicação disponíveis ao utilizador, assim como o comportamento dos dispositivos e do sistema dos vários tipos de dispositivos DP, são especificadas na interface do utilizador.

Todas as variantes, nomeadamente *Fieldbus Message Specification* (FMS), *Decentralized Peripherals* (DP) e *Process Automation* (PA) usam a mesma camada de ligação de dados. E as versões DP e PA utilizam a mesma camada física (ver figura 3.3).

O PROFIBUS dá garantias necessárias para aplicações com transmissão de dados em alta velocidade e com tarefas complexas e extensas de comunicação. Existem várias versões padrão, como o PROFIBUS DP (*master/slave*), o PROFIBUS FMS (*multi-master/peer-to-peer*) e o PROFIBUS PA (intrinsecamente seguro).

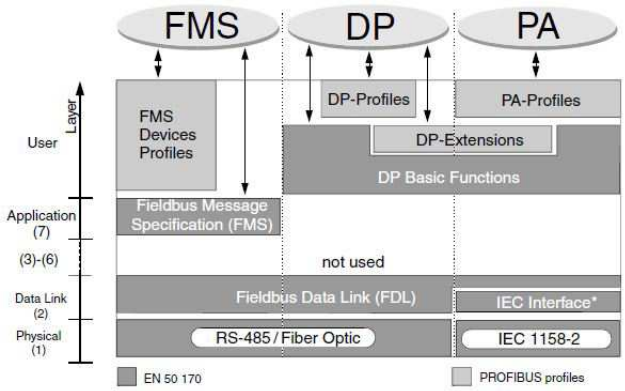

Figura 3.3: Pilha protocolar do PROFIBUS [34].

3.3 Modbus RTU

O protocolo Modbus foi desenvolvido pela Modicon Industrial Automation Systems, nos anos 70. Em 2004, este protocolo tornou-se público a todos os equipamentos, de todas

as marcas, *General Public License* (GPL) [36]. Usualmente é utilizado para trocas de informações, entre:

- dois aplicativos de dispositivos;
- a aplicação do dispositivo e outro dispositivo;
- HMI/*Supervisory Control and Data Acquisition* (SCADA) e dispositivos;
- um computador e um programa de dispositivo que fornece serviços *on-line*.

O protocolo Modbus não define o tipo de ligações físicas entre equipamentos, o tipo de sinais elétricos, óticos ou outros, por isso para transmitir as mensagens tem de ser implementado também um protocolo de comunicação como o RS232, RS485 e Ethernet. Define um simples protocolo *Protocol Data Unit* (PDU) independente das camadas subjacentes de comunicação, de forma a que um conjunto de mensagens, que todos os equipamentos que respeitem este protocolo, saibam interpretar e executar. Dependendo do protocolo utilizado para a troca de dados alguns campos podem ser acrescentados (ver figura 3.4).

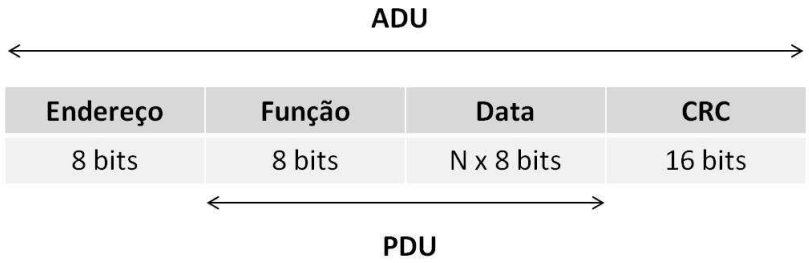

Figura 3.4: Estrutura da mensagem MODBUS-RTU.

O protocolo Modbus usa o tipo de comunicação mestre/escravo, e pode ser inserido numa topologia *unicast* ou *broadcast*. O tipo de diálogo imposto pretende evitar a ocorrência de colisões de dados, quando dois ou mais equipamentos tentarem enviar dados em simultâneo. Num diálogo deste tipo só um equipamento assume o papel de mestre e só ele pode enviar dados para todos os outros sempre que quiser. Os outros equipamentos atuam como escravos, que só podem enviar dados como respostas a um pedido prévio de um mestre.

As mensagens de um dispositivo mestre podem ser de dois tipos, de leitura de entradas, saídas ou memórias, do dispositivo escravo, em que este depois de a receber e a processar responde ao pedido realizado pelo mestre, ou de escrita de dados, em que o escravo limita-se a atualizar as suas posições de memória ou as suas saídas com os valores indicados pelo mestre.

O modelo é baseado por quatro tipos de mensagens:

- Pedido Modbus;
- Indicação Modbus;
- Resposta Modbus;
- Confirmação Modbus;

Um pedido Modbus é a mensagem enviada pela rede do cliente para iniciar uma transação, a indicação Modbus é a mensagem que indica que o pedido foi recebido pelo servidor, uma resposta Modbus é a mensagem de resposta enviada pelo servidor e por sua vez a confirmação Modbus é a mensagem que indica que a resposta foi recebida pelo cliente.

Na mensagem de consulta, o código de função informa ao dispositivo escravo com o respetivo endereço, qual a ação a ser executada. Os bytes de dados contêm informações para o escravo, por exemplo, qual o registo inicial e a quantidade de registos a serem lidos. O campo de verificação de erro permite ao escravo validar os dados recebidos. Na mensagem de resposta, o código de função é repetido de volta para o mestre. Os bytes de dados contêm os dados recolhidos pelo escravo ou o seu estado. Se ocorrer um erro, o código de função é modificado para indicar que a resposta é uma resposta de erro e os bytes de dados contêm um código que descreverá o erro. A verificação de erro permite ao mestre validar os dados recebidos.

3.4 EtherCAT

O *Ethernet Control Automation Tecnhonogy* (EtherCAT) é um protocolo de comunicação industrial de alto desempenho e determinístico para as redes Ethernet. Desenvolvido pela Beckhoff, hoje em dia é aberto e atualizado pelo EtherCAT tecnology group. Expande o padrão IEEE 802.3 para transferir dados com temporização previsível e sincronização precisa. Este padrão aberto foi publicado como parte da norma IEC 61158 e é comumente usado em aplicações como projeto de máquinas e controlo de motores. Estabelece novos limites para o desempenho em tempo-real, uma vez que pode processar 1000 entrada e saídas distribuídas em 30 ms ou 100 eixos em 100 ms usando par traçado ou cabo de fibra ótica. Quanto topologia, EtherCAT suporta uma simples estrutura de baixo custo da linha, uma estrutura de árvore, encadeamento ou linhas de queda - sem componentes de infraestrutura caros [37].

O método de transmissão é semelhante ao praticado pelo protocolo Interbus. Uma vez que a mensagem EtherCAT compreende os dados de diversos dispositivos, os dispositivos escravos EtherCAT lêem os dados endereçados para eles, quando a mensagem passa por eles. As mensagens podem sofrer apenas um atraso na ordem dos nanosegundos. As características *full-duplex* de 100BaseTX são totalmente utilizadas, de modo que as taxas de transmissão sejam superiores a 100 Mbit/s.

O protocolo EtherCAT é otimizado para dados de processo e é transportado diretamente por uma trama Ethernet, graças a um *Ethertype* especial, como clarifica a figura 3.5. Pode consistir em várias sub-mensagens, cada um com uma área de processamento de imagens lógicas que podem ser de até 4 gigabytes de memória particular. A sequência de dados é independente da ordem física dos terminais Ethernet na rede e o endereçamento pode estar em qualquer ordem. Entre escravos a comunicação pode ser do tipo *broadcast*, *multicast* e *unicast*.

Suporta as topologia de linha, de árvore e a de estrela. O meio físico é composto por cabos entraçados Ethernet 100TX com conectores RJ45, que permite uma comunicação a um distância no máximo de 100 metros entre dois nós a comunicar e pode ter no máximo 65535 estações na rede.

A comunicação de dados entre o mestre e o escravo é realizada na forma de objetos

Process Data Objects (PDO). Cada PDO tem o endereço do escravo em particular ou de vários escravos. Uma trama Ethernet pode conter múltiplos desses telegramas, e múltiplas tramas podem ser necessárias para garantir todos os telegramas requeridos para um ciclo de controlo. A tecnologia EtherCAT supera as limitações do sistema ao processar cada frame Ethernet. Quando um dispositivo encontra o pacote Ethernet enviado pelo mestre, ele automaticamente começa a fluir o pacote para outro dispositivo. Como o pacote continua a ser passado de escravo em escravo, ele pode existir em múltiplos dispositivos ao mesmo tempo. Significa desta forma que, por exemplo, se tivermos uma rede com 50 dispositivos escravos, dados diferentes podem ser enviados para cada escravo, sem que seja necessário mandar 50 pacotes diferentes. Contudo se todos os escravos precisarem dos mesmos dados, um pacote mais pequeno poderá ser enviado para todos, otimizando desta forma a velocidade e a largura de banda da transferência de dados.

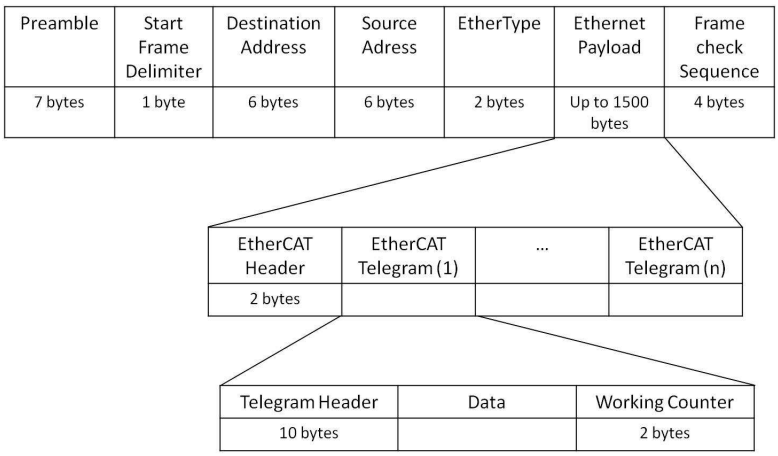

Figura 3.5: Trama EtherCAT (Adaptado a partir [38]).

3.5 Ethernet Powerlink

O Ethernet Powerlink é um protocolo determinístico de tempo-real baseado nas redes industriais Ethernet. Foi inicialmente introduzido pela empresa de automação B&R, em 2001, e em 2002, foi fundado o grupo Ethernet Powerlink Standardization Group (ESPG), que é atualmente responsável pela sua manutenção. Em 2003, foi apresentada a segunda versão que apresenta a camada de aplicação, como extensão à primeira versão. Esta camada é baseada nos mecanismos definidos pelo protocolo CANopen, mantendo a máxima compatibilidade com dispositivos Ethernet existentes [39].

Baseada na camada física Ethernet apresenta grande flexibilidade na topologia a adotar, uma vez que esta pode ser em linha, barramento, estrela ou em árvore, com taxas de transmissão que podem atingir os 100 Mbit/s, e pode ter no limite 240 dispositivos na rede, incluindo o mestre.

Para evitar colisões e maximizar a utilização da largura de banda, é utilizado um esquema de *Time Division Multiple Access* (TDMA), neste caso denominado por *Time Slicing*. O tempo de acesso ao barramento é dividido em "ciclos Powerlink", e por sua vez cada um destes ciclos é dividido em pequenas partes. Assim cada nó apenas pode transmitir no tempo que lhe é reservado.

3.6 PROFINET

O *PROcess Field NET* (PROFINET) é um protocolo aberto para redes industriais Ethernet, desenvolvido pela Siemens e pela *Profibus User Organization* (PNO), e normalizado pelo IEC 61158 e IEC 61784. O conceito PROFINET foi definido em estreita colaboração com utilizadores finais e com base no padrão Ethernet, de acordo com a norma IEEE802. Somente foram especificados acréscimos ao padrão Ethernet nos quais este não podia atender. A modularidade das funções PROFINET torna-o uma solução flexível para todas as aplicações e mercados. É aplicado na automação de sistemas de produção, aplicações com segurança funcional e toda a gama de tecnologias de acionamentos, incluindo aplicações de controlo de movimento isócrono.

O PROFINET resulta dos anos de experiência com o PROFIBUS e da difusão das redes Ethernet. Utiliza também o *Transmission Control Protocol/Internet Protocol* (TCP/IP) e padrão IT, complementando-os com protocolos e mecanismos específicos para atingir um bom desempenho em tempo-real.

Este protocolo apresenta elevada flexibilidade na topologia adaptada, uma vez que pode ser em linha, barramento, estrela, ou em árvore. O PROFINET mantém o padrão Ethernet para o meio físico, possibilitando desta forma adotar configurações mistas de fibra, cabo RJ45 ou M12, e rádio.

O PROFINET oferece um desempenho escalável, com três níveis de desempenho:

- TCP/IP: para aplicações que não tenham requisitos temporais críticos;
- *Real Time* (RT): para trocas de informação em tempo real;
- *Isochronous Real Time* (IRT): para aplicações de controlo de movimento.

O conceito inerente a esta tecnologia propõe uma abordagem modularizada do sistema, dando ao utilizador a possibilidade de criar sistemas complexos de um forma estruturada e sistemática. Desta forma, aborda um série de conceitos como automação distribuída, através do PROFINET CBA, e dispositivos de campo descentralizados, através do PROFINET IO.

O PROFINET CBA é baseado na modelação orientada a objetos tecnológicos modulares. A funcionalidade de um módulo é encapsulada no mesmo, podendo ser vista como uma "caixa negra"pelo sistema, que é acedido através de uma interface. Desta forma, é possível criar sistemas compostos, efetuando a interligação de vários módulos.

O PROFINET IO permite a utilização de dispositivos de campo descentralizados sobre Ethernet. Define um acesso de forma homogénea aos dispositivos, especificando o método de troca de dados entre controladores e dispositivos. Segue uma filosofia produtor/consumidor, apresentando uma troca de dados de elevada velocidade. Ou seja, este utiliza por isso as comunicações RT e IRT. As comunicações RT apresentam ciclos de transmissão da ordem do milissegundos sendo adequadas para a ligação a periféricos. As comunicações IRT apresentam também ciclos de transmissão na ordem dos milissegundos, apropriadas para aplicações de controlo de movimento.

3.7 Modbus/TCP

O Modbus/TCP é uma rede Ethernet industrial aberta que foi especificada pela Organização Modbus-IDA, em cooperação com a *Internet Engineering Task Force* (IETF).Com

a grande utilização e pela facilidade na modificação do protocolo Modbus, foram incorporados novos meios de comunicação como as redes Ethernet, que são as redes mais utilizadas a nível mundial, dominantes ao nível de redes locais e do acesso à Internet.

O Modbus estrutura as mensagens definindo algumas regras de envio e interpretação e o TCP/IP permite que os blocos de dados binários sejam trocados entre equipamentos, provendo desta forma o meio transporte das mensagens. Em resumo, o Modbus TCP/IP combina uma rede física Ethernet, com uma rede padrão TCP/IP, e um método padrão de dados que representam como Modbus o protocolo de aplicação.

Como foi anteriormente referido, o Modbus é um protocolo baseado num modelo pedido/resposta oferecendo serviços especificados por funções com códigos. Os códigos das funções Modbus fazem parte da mensagem PDU, tal como acontece com o Modbus-RTU. As diferenças para o Modbus-RTU vão estar na parte *Application Data Unit* (ADU) da mensagem (ver figura 3.4).

Tal como as outras redes baseadas em Ethernet, este protocolo é muito flexível na topologia a ser adotada e permite ter as taxas de transmissão padrão Ethernet, de 10, 100 ou 1000 Mbit/s, através dos meios físico por cabo, fibra ótica ou *wireless*.

3.8 CANopen

O protocolo CANopen foi pré-desenvolvido num projeto designado por Espritm durante a presidência da Bosch. Em 1995, a especificação do CANopen foi passada para a *CAN in Automation* (CiA), um grupo internacional de utilizadores e fabricantes. Originalmente, o perfil de comunicação CANopen era baseado no protocolo da camada de aplicação.

A versão 4 do CANopen, CiA *Draft Standard* (DS) 301, está padronizada como EN50325-4. A especificação CANopen abrange a camada de aplicação e perfil de comunicação, CiA DS 301, assim como a estrutura para equipamentos programáveis CiA DS 302, recomendações para cabos e conetores CiA DS 303-1 e representações de prefixos e unidades SI, CiA 303-2, especificações estas publicadas em [40]. A camada de aplicação, assim como os perfis baseados em CAN, são implementados através do *software*.

A arquitetura de uma rede é formada por camadas, interfaces e protocolos. As camadas são processos, implementados por *hardware* ou *software*, para permitir a comunicação entre máquinas. Cada camada proporciona um conjunto de serviços ao nível superior, através das funções que detém na própria camada e das camadas inferiores disponíveis. Como ilustra a figura 3.6, o CANopen é um protocolo de alto nível, que está posicionado na última camada do modelo de referência OSI, implementando o protocolo CAN para as duas primeiras camadas.

O modelo prevê uma separação clara entre essas camadas com interfaces definidas. Como anteriormente foi mencionado, uma aplicação com todas as camadas envolvidas não seria adequada para sistemas integrados, por causa da sobrecarga de todas estas interfaces, que dificilmente seria vantajoso implementar sistemas de comunicação eficientes para microcontroladores de gama baixa e com recursos limitados.

Algumas aplicações colocam a sua própria camada de aplicação diretamente sobre a camada de ligação de dados. No entanto, o CANopen aplica pelo menos partes de outras camadas (ver figura 3.6) [41]:

1. Camada física:
   - Descreve a interligação física entre os nós da rede (especifica o uso da ISO 11898, de alta velocidade);
   - Inclui características elétricas dos sinais usados (o *transceiver* escolhido utiliza sinal diferencial);
2. Camada de ligação de dados:
   - Bits são combinados em *frames*;
   - Inclui deteção de erros através de *checksums*;
   - Define o conjunto de reconhecimentos para determinar se a transmissão foi bem sucedida;
3. Camada de rede:
   - Inclui conceitos do destino endereçamento e encaminhamento;
   - Fornece a funcionalidade de interação entre um *host* e a rede (CANopen: configuração via *Service Data Objects* (SDO));
   - Utiliza a fragmentação para permitir a transmissão de mensagens maiores do que o disponível pelo campo de dados de 8 bytes;
4. Camada de transporte:
   - Fornece a fiabilidade *end-to-end*: comunicação entre a origem e o destino (parcialmente prestado pelo serviço *Network Management* (NMT));
5. Camada de sessão:
   - Sincronização: Pode ser usado para grandes transferências de dados - pode retomar uma transferência interrompida;
6. Camada de apresentação:
   - Identificadores para acesso ao dados e codifica os dados de uma forma padronizada (CANopen: Dicionário de objetos);
7. Camada de aplicação:
   - Programas de aplicação que usam a rede.

O perfil CiA DS 301 é a especificação fundamental do CANopen, onde existe um grande número de outros documentos que definem os dispositivos padrão ou aplicações normais (ver figura 3.7). Nessas normas complementares, o comportamento e os parâmetros de dispositivos ou aplicações são definidas pelas entradas do dicionário de objetos correspondentes. O protocolo também define perfis de aplicação que devem facilitar a integração de sistemas, compreendendo dispositivos de diferentes fornecedores. De uma maneira simples, os perfis de dispositivos genéricos descrevem a interface de um único dispositivo, enquanto os perfis de aplicação descrevem todas as interfaces dos dispositivos que fazem parte de uma aplicação. Perfis de dispositivos também podem conter códigos adicionais de erro, tipos de dados compilados, LEDs de dispositivos, entre outras especificações.

O CiA 401, designado por perfil do dispositivo para módulos I/O, é o perfil de dispositivo mais conhecido e mais importante. Este perfil descreve entradas digitais e analógicas

e interfaces de saída e a sua capacidade de ser parametrizado. Este perfil do dispositivo especifica as entradas do dicionário de objetos para um máximo de 2040 entradas e/ou saídas digitais e 255 entradas e/ou saídas analógicas. Além das entradas padronizadas para os valores atuais, há uma série de outras entradas do dicionário de objetos para parametrizar o comportamento dessas entradas e saídas.

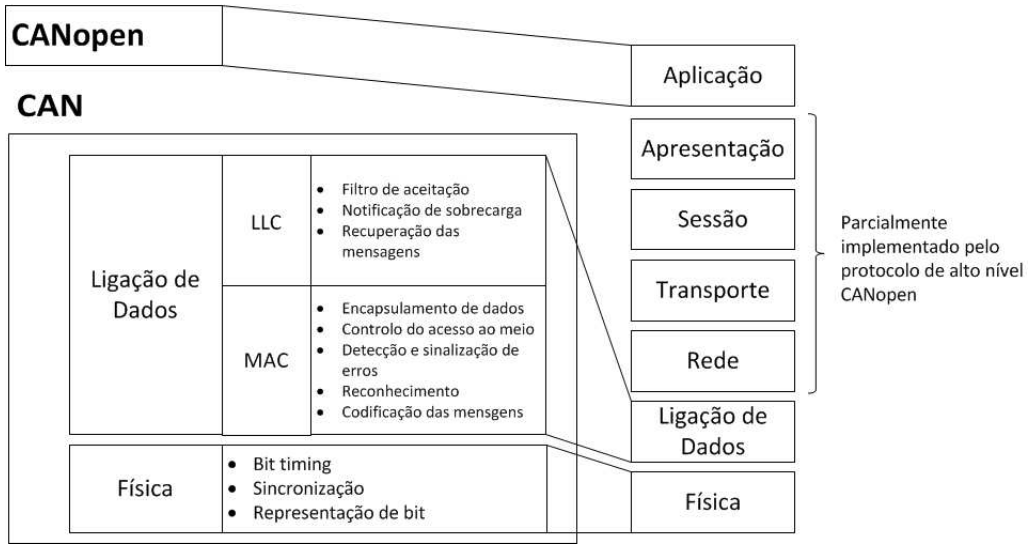

Figura 3.6: CANopen e o modelo de referência OSI (Adaptado a partir de [41]).

Outro perfil do equipamento muito utilizado é o CiA 402, *Drives and Motion Control*. Este perfil pretende abranger o servo controladores, o motores de passo e os conversores de frequência.

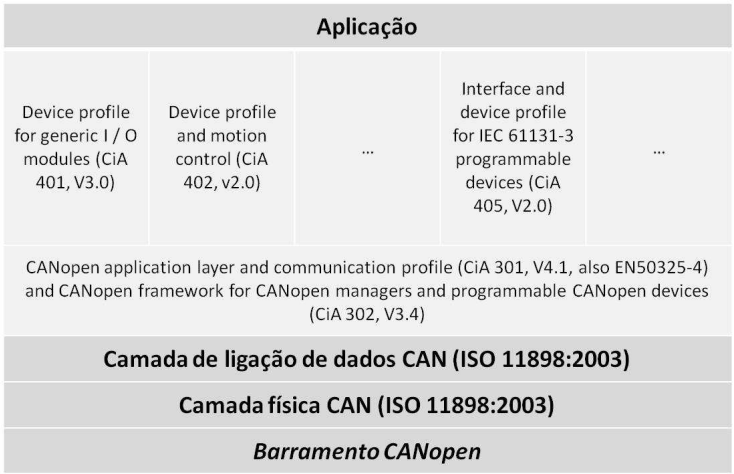

Figura 3.7: Descrição do protocolo CANopen (Adaptado a partir de [42]).

3.8.1 Protocolo subjacente CAN

O protocolo CAN foi originalmente concebido para o setor automóvel, onde vários sensores enviam informação em pequenas quantidades e com elevada frequência. É um protocolo de comunicações que suporta eficientemente o controlo de sistemas distribuídos em tempo-real, com um nível elevado de segurança. O seu domínio de aplicação abrange desde redes de alta velocidade, até 1 Mbit/s, até ligações multiplexadas de baixo custo, sendo por isso economicamente viável para implementar nos dispositivos utilizados no interior dos veículos, permitindo assim eliminar os inúmeros cabos, normalmente usados.

Este protocolo é usualmente implementado num controlador na forma de um circuito integrado, mas também se encontra no mercado microcontroladores de 8, 16 e 32 bits com controladores CAN integrados. Os controladores CAN bem como os microcontroladores com controladores CAN integrados são fabricados por um grande número de indústrias, como Intel, Motorola, Philips, Siemens e Texas Instruments, resultando em torno de 50 circuitos integrados de 15 fabricantes diferentes. Na indústria automobilística resultou numa produção em grande escala de controladores CAN, atualmente na casa dos milhões ao ano [33].

A afirmação e o crescimento do protocolo nos dispositivos industriais, deve-se essencialmente a organizações de fabricantes em torno de associações, como a CiA, assim como na existência de uma série de ferramentas de *software* para o desenvolvimento, simulação, configuração e monitorização de aplicações, na disponibilidade de *hardware* na forma de placas de controlo, placas ISA, PCI e outras [33].

O objetivo da sua especificação é conseguir a compatibilidade entre as várias implementações CAN, deixando de fora a compatibilidade relativa a especificações elétricas e a interpretação dos dados transferidos. Para atingir a transparência e a flexibilidade de implementação, o protocolo CAN é definido em duas camadas diferentes do modelo OSI, a camada física e a camada de ligação dos dados.

Este modelo é composto por sete camadas e está dividido em camadas hierárquicas, ou seja, cada camada usa funções da própria camada ou de camada anterior, para esconder a complexidade e transparecer as operações para o utilizador. A divisão deste modelo em camadas é realizada de forma transparente e durante a comunicação entre dispositivos da rede.

Camada física

A camada física trata de aspetos como a temporização, a codificação de bits, a sincronização dos mesmos e quais os cabos e conetores que serão utilizados na instalação da rede. Uma rede CAN pode ser montada utilizando apenas dois fios, CAN_H (CAN High) e CAN_L (CAN Low). Muitas vezes além destes dois sinais de dados é utilizado mais um fio, o GND (referência) no barramento.

Níveis de Tensão

Os dados enviados são interpretados pela análise da diferença de potencial entre os condutores CAN_H e CAN_L, ilustrado na figura 3.8. Este conceito de diferença de potencial atenua fortemente os efeitos causados por interferências eletromagnéticas, uma vez que qualquer ação exterior será sentida pelos dois condutores, causando flutuação em ambos os sinais para o mesmo sentido e com a mesma intensidade. Nos módulos recetores é a

diferença de potencial entre condutores CAN_L e CAN_H, que permanece inalterável, garante a sua continuidade.

O meio de transmissão no CAN deverá permitir a representação do estado dominante e recessivo. Para os meios elétricos as tensões de saída para o barramento são definidas pela ISO 11898-2, ISO 11898-3, SAE J2411 e ISO 11992.

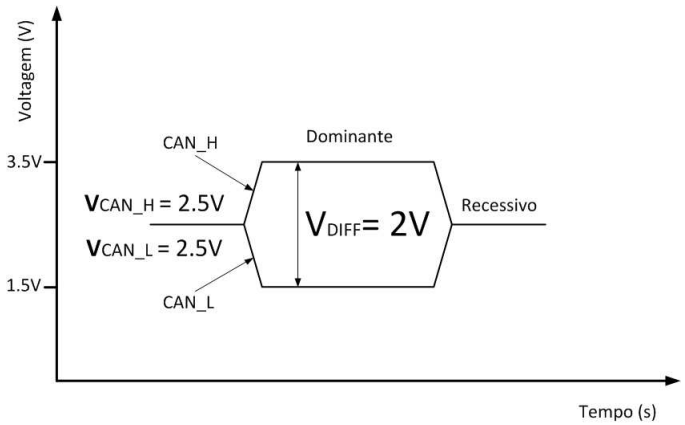

Figura 3.8: Níveis de tensão do barramento CAN.

O sinal diferencial fornece já um bom nível de proteção às interferências eletromagnéticas, e alguns casos com o par de fios entrançados sem blindagem pode ser concebida uma boa solução contudo, a blindagem é recomendada. O meio de transmissão também pode ser a fibra ótica, que permite um isolamento galvânico e uma grande eficiência. Esta solução é interessante para aplicações em ambientes com perigo de explosão ou em ambientes com muito ruído eletromagnético [43].

Existe também uma solução designada por PLC (*Power Line Comunication*) que permite transmitir sinais CAN na linha de alimentação. Este tipo de comunicação é possível através de um componente designado por DCAN250 [44]. Este componente funciona com um *transceiver* CAN com capacidade para utilizar a linha de alimentação para a comunicação entre nós.

Numa rede CAN cada nó opera em sincronia com um oscilador que gera um sinal de frequência pré-programada. Cada conjunto de n períodos dessa frequência é designado por *Time-quantum* (Tq), sendo o tempo de transmissão de cada bit definido como múltiplo de Tq.

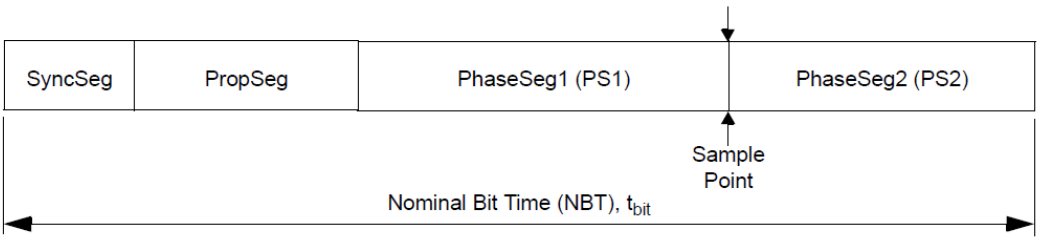

Figura 3.9: Segmentos da mensagem CAN [45].

Como se pode verificar na figura 3.9, existe um segmento de sincronização com uma duração de um Tq, um segmento de propagação com duração de um a oito Tq e dois segmentos de fase de um a oito Tq cada. O comprimento dos segmentos de propagação e de fase são programáveis, no qual os segmentos de fase são reajustáveis durante as ressincronizações. No total o bit-time dura entre 8 a 25 Tq [45].

O segmento de sincronização é utilizado para uma sincronização inicial. O segmento de propagação prevê a existência de um atraso de propagação e de processamento entre os vários nós na rede, para compensar alguns atrasos.

Camada de ligação de dados

A camada de ligação de dados é responsável pela construção das mensagens de dados, pela sua manipulação e também por realizar os controlos na transmissão. Esta camada é responsável pela identificação dos pacotes de dados, pelo controlo do acesso ao barramento e também pelas verificações de possíveis erros nas transmissões ou no conteúdo das mensagens que são enviadas.

O controlo de acesso ao barramento é realizado através de um método de arbitragem conhecido com *Carrier Sense Multiple Access with Non-destructive Bitwise Arbitration* (CSMA/NBA) com a finalidade de evitar a colisão de dados durante as transmissões entre os dispositivos instalados na rede.

O método de arbitragem CSMA/NBA é baseado no identificador das mensagens para análise das prioridades de acessos. Em caso de disputa pelo barramento entre dois ou mais dispositivos, o dispositivo que possuir o identificador da mensagem com maior prioridade continua a sua transmissão, enquanto que os outros dispositivos que possuam uma mensagem de prioridade mais baixa, vão interromper a sua transmissão e armazenar as suas mensagens em *buffers* de memória, para, posteriormente, quando obtiverem acesso ao barramento, procederem ao seu envio.

O controlo de erros é realizado através da monitorização dos problemas na transmissão das mensagens e armazenamento dos erros ocorridos, por um dado dispositivo, com o propósito de limitar a quantidade de erros por dispositivo. Assim, um dispositivo que exceda um limite de erros em transmissões poderá ser desligado automaticamente da rede.

O identificador com o número menor possui prioridade mais alta e, por consequência, os identificadores com maior número possuem prioridade mais baixa. O tamanho destes identificadores varia consoante a versão: para a versão CAN 2.0A o identificador é composto por 11 bits e para a versão CAN 2.0B são 29 bits.

Com o barramento livre, qualquer dispositivo pode iniciar a sua transmissão de mensagens. Se existir algum conflito, a mensagem com maior prioridade é a primeira a ser transmitida, enquanto as de prioridade mais baixa aguardarão até que o barramento esteja livre, com base no método de arbitragem. O conflito é resolvido pela comparação bit a bit do identificador das mensagens, ou seja, em cada nó que for disputada a transmissão, o bit do identificador é comparado com o bit presente no barramento. Quando um nó transmite um bit recessivo, ou "1"lógico, e no barramento se encontra um bit dominante, ou "0"lógico, este aborta de imediato a sua transmissão e aguarda que o barramento fique livre para iniciar nova transmissão (ver tabela 3.3). Este método de arbitragem garante que não sejam perdidos dados. O método de arbitragem do protocolo CAN é uma forma de resolver conflitos e colisões nos acessos ao barramento, contudo em

sistemas com uma grande quantidade de dispositivos conectados à rede, mensagens com baixa prioridade vão possuir uma latência muito elevada de acesso ao barramento. Nestes casos, a definição dos identificadores das mensagens é uma tarefa bastante importante, pois estes estão diretamente ligados ao tempo de acesso ao barramento.

Utilizando o formato padrão de mensagens é possível obter 2048 mensagens numa rede, o que pode caraterizar uma limitação em determinadas aplicações, enquanto que, no formato estendido é possível obter aproximadamente 537 milhões de mensagens numa rede. No segundo método abordado, já não está presente a limitação em relação ao número de mensagens, mas por outro lado, com o acréscimo de 18 bits no campo identificador, o tempo de cada mensagem vai aumentar, o que pode constituir um problema em determinadas aplicações que trabalhem em tempo-real, problema este designado por *overhead*.

Dispositivo A	0	1	0	0	1	1	1	0	1	0	1
Dispositivo B	0	1	0	0	1	1	1	0	0	0	1
Dispositivo C	0	1	0	1	1	1	1	0	1	0	1
Barramento	0	1	0	0	1	1	1	0	0	0	1

Tabela 3.3: Método de arbitragem de acesso ao barramento CAN [4].

Tramas CAN

O protocolo CAN possui quatro tipos de tramas: trama de dados, trama remota, trama de erro e trama de sobrecarga. A trama de dados é a única das tramas CAN que permite o envio no máximo de 8 bytes por mensagem. O protocolo CAN utiliza este tipo de tramas para transmitir informação entre os dispositivos da rede. Estas tramas encontram-se divididas em sete campos distintos.

O campo *Start of Frame* (SOF) é composto por um bit dominante, tanto no formato padrão como no estendido, que marca o início de uma trama de dados.

No formato padrão, ilustrado pela figura 3.10, o identificador é composto por 11 bits. Este campo é responsável pelo acesso ao barramento, seguido do bit *Remote Transmission Request* (RTR), que identifica o tipo de trama. Numa trama de dados o bit RTR é dominante e numa trama remota o RTR é recessivo.

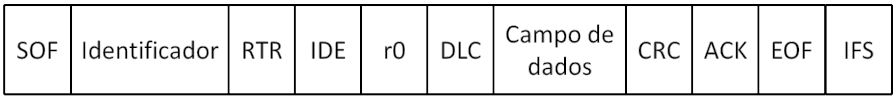

Figura 3.10: Formato de uma trama de dados padrão.

No formato estendido, ilustrado pela figura 3.11, o identificador é composto por 29 bits, ID base (11 bits) + ID estendido (18 bits). O bit *Substitute Remote Request* (SRR) garante maior prioridade para mensagens com formato padrão em relação ao formato estendido, desde que ambas tenham o mesmo identificador base. O bit *Identifier Extension* (IDE) distingue a trama padrão, se o bit for dominante, da trama estendida, se o bit for recessivo. O bit RTR tem o mesmo significado das tramas padrão.

O campo de controlo é um campo constituído por 6 bits, quer para tramas padrão que para tramas estendidas. A única diferença encontra-se no primeiro bit. Nas tramas

Figura 3.11: Formato de uma trama estendida.

padrão é o bit IDE que deve ser dominante, e nas tramas estendidas é o bit r1 que também deve ser dominante. Segue-se, em ambos os formatos, o bit reservado r0, que deve ser enviado como dominante, seguindo-se de quatro bits que compõem o *Data Length Code* (DLC), que indica o tamanho do campo de dados em bytes, que pode variar de 0 a 8 (ver tabela 3.4).

Byte	**DLC3**	**DLC2**	**DLC1**	**DLC0**
0	D	D	D	D
1	D	D	D	R
2	D	D	R	D
3	D	D	R	R
4	D	R	D	D
5	D	R	D	R
6	D	R	R	D
7	D	R	R	R
8	R	D	D	D
D = Bit Dominante		**R = Bit Recessivo**		

Tabela 3.4: Valores aceitáveis no campo DLC [33].

O campo de dados pode conter 0 a 8 bytes. Sem possuir um tamanho fixo, terá sempre que enviar em múltiplos de 1 byte e até ao limite de 8 bytes.

O campo *checksum* contém a sequência de *Cyclic Redundancy Check* (CRC), composto por 15 bits, que permite detetar se os dados de uma mensagem CAN foram recebidos sem erros. Depois destes 15 bits segue-se o CRC *Delimiter*, composto apenas por um bit recessivo.

O campo de reconhecimento é composto por dois bits, o primeiro ACK SLOT e o segundo ACK *Delimiter*. O dispositivo que enviar uma mensagem coloca estes dois bits como recessivos, qualquer dispositivo que receba a mensagem reescreve o bit ACK SLOT como dominante, sinalizando assim ao emissor qual a mensagem foi recebida pelo menos por um dispositivo.

O campo de fim de trama *End of Frame* (EOF), tanto para o formato padrão como para o estendido, é composto por 7 bits recessivos, que indica o fim da trama CAN.

A trama remota é utilizada para a solicitação de informação por um dispositivo conetado a uma rede CAN. Apresenta um formato semelhante ao anterior, porém durante o envio de uma mensagem, a trama remota não apresenta campo de dados. O bit RTR nas tramas remotas é enviado como recessivo.

A trama de erro é enviada por um dispositivo para todos os outros dispositivos, quando é detetado um erro numa comunicação, indicando que foi detetado um erro na transmissão. Após a receção desta mensagem o dispositivo que estava a transmitir vai iniciar a retransmissão. A trama de erro é composta por dois campos: o campo de erro e o campo delimitador. O campo de erro viola o *Bit Stuffing* enviando uma sequência de 6

bits, dominantes ou recessivos, consecutivos. O campo delimitador é composto por 8 bits recessivos que marcam a terminação de uma trama de erro. Há dois tipos de tramas de erro: tramas de erro ativas, composta por 6 bits dominantes e as tramas de erro passivas, composta por 6 bits recessivos. Ambas violam a regra do *Bit Stuffing*, regra que permite que sejam enviados somente 5 bits consecutivos com o mesmo valor.

As tramas de sobrecarga são semelhantes às tramas de erro. As tramas de sobrecarga são geralmente utilizadas com o objetivo de atrasar o próximo envio de uma trama de dados, ou uma trama remota, evitando assim erros nas transmissões. Este tipo de trama pode ser enviado quando um dispositivo recetor necessitar de atrasar o envio da próxima mensagem, obtendo assim mais tempo para processar a mensagem atual. As tramas de sobrecarga são compostas por dois campos: *flag* de sobrecarga e o delimitador de sobrecarga. A *flag* de sobrecarga é composta por 6 bits dominantes e o delimitador de sobrecarga é composto por 8 bits recessivos.

O espaçamento entre tramas corresponde ao período de tempo que separa a transmissão consecutiva de duas tramas (nas tramas de dados ou tramas remotas). Desta forma, sempre que um nó quiser transmitir uma trama de dados ou remota, deverá aguardar até que detete o espaçamento entre tramas, para iniciar a transmissão. Pode-se então dizer que as tramas de dados e remotas, são sempre precedidas de um espaçamento entre tramas, o mesmo não se verifica nas tramas de erro e de sobrecarga. O espaçamento entre tramas é composto por dois campos fixos: o campo de intervalo, *Intermession*, e o barramento livre, *BusIdle*. O campo de intervalo é composto por 3 bits recessivos após o campo EOF, enquanto que o barramento livre não possui tamanho fixo, sendo constituído por uma sequência de bits recessivos, até que seja detetado um bit dominante no barramento, que simboliza o início de uma trama SOF.

A implementação do controlador CAN possui um mecanismo de filtragem de mensagens baseado no identificador das mesmas. Com este mecanismo, os dispositivos conetados a uma rede, só processarão as mensagens que passem pelo filtro, ignorando as restantes mensagens. É possível receber mensagens com base no identificador completo ou então mascarar o identificador, ou seja, escolher apenas os bits do identificador que se quer filtrar, permitindo assim que um dispositivo receba determinados grupos de mensagens. O conjunto filtro/máscara atua da seguinte forma, no filtro são definidos os identificadores que se pretende receber e na máscara são definidos os bits que se pretende comparar com os do identificador (ver tabela 3.5).

Filtro	0	1	0	1	1	0	0	0	0	1	1
Máscara	1	1	1	1	0	0	0	0	0	0	0
Ids rececionados	0	1	0	1	x	x	x	x	x	x	x

Tabela 3.5: Processo de filtragem do barramento CAN [4].

3.8.2 Dicionário de Objetos e *Electronic Data Sheet*

A descrição padronizada denominada por dicionário de objetos é uma das propriedades mais importantes deste protocolo. Apresenta um grupo de objetos acessíveis através da rede numa maneira ordenada pré-definida e representa o acesso completo para o programa de aplicação dos equipamentos tanto dos dados de aplicação como dos parâmetros de

configuração. Assim, é possível ter acesso a todos os dados importantes, parâmetros e funções de um dispositivo, usando um sistema de endereçamento lógico, com um índice de 16 bits e um subíndice de 8 bits.

O dicionário de objetos não só fornece uma maneira de associar variáveis com um valor de índice e subíndice, mas também especifica uma tabela de definição de tipos de dados. A tabela 3.6, mostra as sete primeiras entradas do dicionário de objetos que define alguns tipos de dados. As entradas acima mencionadas são apenas usadas para definir os tipos de dados e não para armazenar todas as variáveis. Se uma especificação diz que um nó deve ter uma variável chamada "Posição em X", que está localizado no índice 2000h, subíndice 0 e o tipo de dados for 4, de acordo com o dicionário objetos o tipo de dados é INTEGER32, um valor inteiro de 32 bits [41].

Índice	**Tipo de dados**
1	BOOLEAN
2	INTEGER8
3	INTEGER16
4	INTEGER32
5	UNSIGNED8
6	UNSIGNED16
7	UNSIGNED32

Tabela 3.6: Tipos de dados que definem as entradas do dicionário de objetos.

Como foi referido anteriormente, além da descrição padronizada das propriedades de comunicação dos dispositivos de acordo com a CiA 301, o CANopen define os chamados "perfis de dispositivos"para dispositivos típicos de áreas de aplicação distintas. Estes perfis especificam os parâmetros mais importantes, dados e funções por tipo de dispositivo, por exemplo, módulos com entradas/saídas, unidades, *encoders*. Habitualmente os dispositivos com este protocolo implementado recorrem de uma folha de dados eletrónica *Electronic Data Sheet* (EDS), que contém o tipo de dados e a função de cada entrada do diretório. Normalmente, a EDS é um arquivo ASCII, contendo todos os dados. Do ponto de vista da comunicação, quaisquer dois nós que dispõem a mesma EDS são permutáveis, os seus dicionários de objetos são idênticos e têm o mesmo comportamento de comunicação.

Índice	Objectos
0	Não é usado
0001 - 001F	Static Data Types
0020 - 003F	Complex Data Types
0040 - 005F	Manufacturer Specific Complex Data Types
0060 - 007F	Device Profile Static Data Types
0080 - 009F	Device Profile Specific Complex Data Types
00A0 - 0FFF	Reservado
1000 - 1FFF	Communication Profile Area
2000 - 5FFF	Manufacturer Specific Profile Area
6000 - 9FFF	Standardised Device Profile Area (DSP-401)
A000 - FFFF	Reservado

Tabela 3.7: Estrutura genérica do dicionário de objetos CANopen.

3.8.3 *Network Management*

O serviço NMT do CANopen segue uma estrutura mestre/escravo. Requer que um dispositivo na rede desempenhe as funções de NMT *master* e que os outros dispositivos desempenhem a função de NMT *slave*, ilustrado pela figura 3.12.

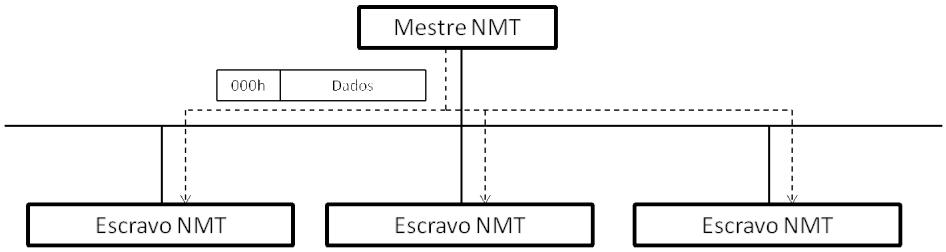

Figura 3.12: Esquema da comunicação NMT (Adaptado a partir de [42]).

A gestão da rede fornece as seguinte funcionalidades:

- Serviço de inicialização de dispositivos NMT *slave* que participem na rede;
- Serviço de controlo de erros para supervisão dos dispositivos;
- Gestão do estado de comunicação da rede;
- Serviço de controlo da configuração dos dispositivos.

A mensagem NMT é a mensagem com maior prioridade, que transmitida pelo NMT *master* força a que os outros dispositivos conetados à rede transitem para outro estado.

Os objetos 1F80h e 1F81h são definidos em CiA 302, como o quadro de gestores CANopen. O dispositivo NMT *master* indica no dicionário de objetos no índice 1F80h.

Os estados que são especificados pelo mecanismo NMT do CANopen são os estados de inicialização, pré-operacional, operacional e parado. Na figura 3.13, são ilustrados a interação entre os diferentes estados, tais como os comandos responsáveis pelas respetivas transições de estados. Mais à frente será explicado, como este comando pode ser enviado para os dispositivos.

	Inicialização	Pré-operacional	Operacional	Parado
Boot-up	X			
SDO		X	X	
EMCY		X	X	
Sync		X	X	
Heartbeat		X	X	X
PDO			X	

Tabela 3.8: Comunicação dependente dos estados NMT (Adaptado a partir de [41]).

Depois de ligado um equipamento CANopen, este passa pelo estado de inicialização e automaticamente passa para o estado pré-operacional (ver tabela 3.8 e figura 3.13) . Neste estado o nó vai:

- permitir a comunicação via SDO para, por exemplo, configurar o mapeamento PDO e parâmetros de comunicação;
- mudar para o estado operacional se receber o comando *Start Remote Node*;
- mudar para o estado preparado se receber o comando *Stop Remote Node*;
- reiniciar *firmware* ou a comunicação, se receber o comando *Reset Node* ou *Reset Communication*.

No estado operacional o dispositivo vai:

- puder comunicar via SDO e PDO;
- enviar mensagens de emergência na ocorrência de algum erro;
- mudar para o estado preparado se receber o comando *Stop Remote Node*;
- mudar para o estado pré-operacional se receber o comando *Enter Pre-Operational State*;
- reiniciar *firmware* ou a comunicação,se receber o comando *Reset Node* ou *Reset Communication*.

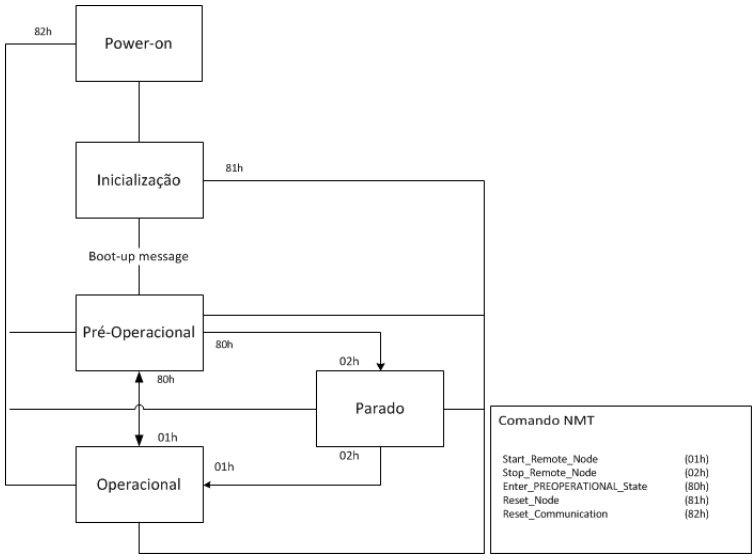

Figura 3.13: Estados NMT.

No estado parado o nó desativa quase todos os serviços. O nó ficará neste estado a aguardar a receção de uma mensagem NMT com um comando para mudar para outro estado. É importante que seja referido que alguns destes podem ser acessíveis só para leitura, por causa de alguns aspetos de implementação. Por exemplo, se o nó estiver no estado operacional e não seja adequado que um objeto que contenha o programa da aplicação possa sofrer alterações durante a sua execução.

3.8.4 *Service Data Objects*

Os objetos de comunicação específicos, chamados SDOs, são usados para o acesso direto aos dispositivos CANopen. A partir da comunicação SDO, as entradas do dicionário de objetos podem ser lidas e escritas, numa comunicação que ocorre sempre por uma conexão lógica 1:1 entre dois nós, por exemplo, entre um nó de configuração e um nó a ser configurado. O comprimento da informação que pode ser transmitida através da comunicação SDO é teoricamente ilimitada (ver figura 3.14).

Como a transferência de dados é efetuada através de uma ligação com um serviço reconhecido, significa que um pedido realizado pelo dispositivo cliente terá sempre uma resposta por parte do dispositivo servidor, mesmo que o dispositivo não seja capaz de fornecer os dados ou o próprio pedido constitua um erro. Se o pedido não for atendido para além do código de erro de 4 bytes de comprimento, que especifica a causa da falha, também refere qual a entrada do dicionário de objetos para o qual a transferência SDO teve insucesso. Esta característica pode ser muito útil, especialmente na configuração de um dispositivo.

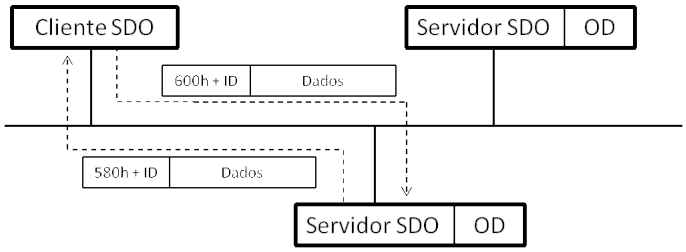

Figura 3.14: Esquema de comunicação SDO (Adaptado a partir de [42]).

3.8.5 *Process Data Objects*

O serviço de comunicação PDO é um serviço eficiente de transmissão de informação de partes do dicionário de um dispositivo CANopen. Uma das principais tarefas de um sistema CANopen é a troca de dados de processo. Na transmissão de dados de processo, como ilustra a figura 3.15, a transmissão pode ser descrita através do modelo produtor/consumidor, isto é, a mensagem é transmitida por um produtor e mais que um dispositivo consumidor poderá a receber. Portanto trata-se de uma comunicação adequada para transmissão de dados em tempo-real. No entanto, tem que ser considerado que neste tipo de comunicação os dados são transmitidos sem confirmação da sua receção. Significa que não há reconhecimento de que a informação foi recebida por qualquer participante do barramento.

A transmissão do PDO só é possível no estado operacional. O campo de dados utilizado pela comunicação PDO poderá conter múltiplos de um byte e no máximo oito bytes de dados. O seu conteúdo não é facilmente interpretado, porque a ideia básica é que o recetor saiba como poderá o conteúdo da mensagem PDO ser interpretado. Por esta razão, este serviço de comunicação requer configuração do utilizador, para que depois a comunicação seja gerida pelo *software* que implemente o protocolo. A configuração passa por definir o identificador da mensagem CAN a ser usado na transmissão, do tipo de acionamento da mensagem e dos objetos que serão incluídos no campo de dados da mensagem CAN (ver figura 3.16). O produtor transmite a mensagem PDO e compõe

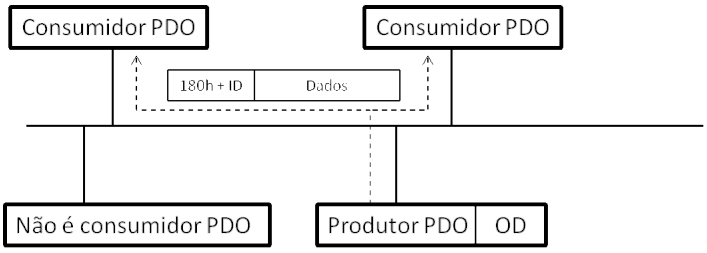

Figura 3.15: Esquema de comunicação PDO (Adaptado a partir de [42]).

o campo de dados, de acordo com a configuração que tem armazenada no dicionário de objetos, mais precisamente nos parâmetros de transmissão PDO e nos parâmetros de mapeamento PDO. O mesmo acontece no lado do consumidor, em que a partir dos parâmetros de receção PDO e dos parâmetros de mapeamento PDO do seu dicionário, os bytes de dados das PDO recebidas são estruturados e armazenados no seu dicionário de objetos. Os nós consumidores selecionam as mensagens a partir do *Communication Object Identifier* (COB-ID) definido nos parâmetros de receção PDO.

Um equipamento que suporte as variáveis mapeadas de PDOs deve suportar as mesmas durante o estado pré-operacional. Se for suportado o mapeamento dinâmico durante o estado operacional, o cliente SDO é responsável pela consistência dos dados.

As transmissões PDO podem ser orientadas por um evento ou temporizador interno, por uma requisição remota ou por uma mensagem de sincronização recebida:

- orientado por evento: um evento especificado no perfil do equipamento, causa a transmissão da mensagem. Com um período de tempo passado faz com que os nós também transmitam;
- requisição remota: outro equipamento pode inicializar a transmissão de um PDO assíncrono pelo envio de uma requisição remota;
- transmissão sincronizada: na transmissão síncrona o nó aguarda uma mensagem Sync para transmitir uma mensagem PDO.

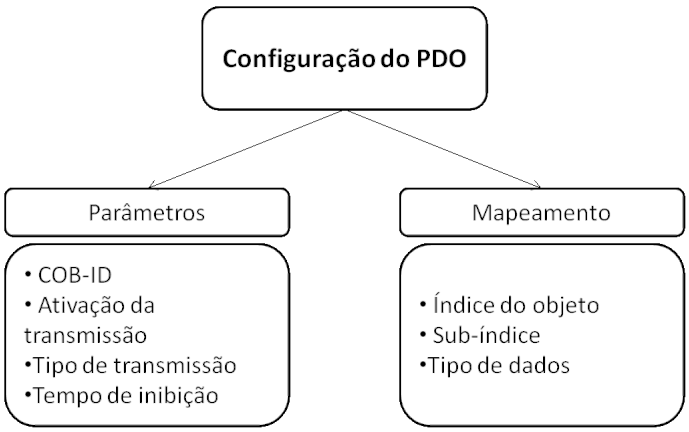

Figura 3.16: Configuração do serviço de comunicação de tempo crítico.

Este processo de transferência de dados é um dos elementos diferenciadores desta camada de aplicação o número de combinações possíveis que n dados podem ser empaco-

tados em i PDOs é k_i^n. O número de combinações que n dados podem ser colados numa PDO pode ser calculado pela equação [46]:

$$k^n = 2 + \sum_{i=2}^{n-1} k_i^n \tag{3.1}$$

Os autores [47; 48], apresenta um estudo com cinquenta e três sinais foram mapeados em 17 mensagens CAN usando o serviço PDO, foram testados os tempos de resposta nos piores casos com uma taxa de transmissão de 125 kbit/s. Neste cenário que pretende verificar o pior dos casos, todas as mensagens cumpriram os seus prazos e registaram uma percentagem de ocupação do barramento de 84,44%. Este estudo mostra claramente como este serviço poderá ser útil para requisitos em tempo-real e que este reduz substancialmente a sobrecarga da comunicação.

3.8.6 Serviço de Sincronização

Como pode ser ilustrado pela figura 3.17, serviço de sincronização segue um modelo produtor/consumidor. Um único produtor fornece um sinal de sincronização para cada consumidor que necessite deste sinal, como por exemplo, o servidor PDO. Normalmente este serviço é usado para sincronizar a leitura e a escrita no dicionário de objetos utilizando o serviço PDO. Quando é recebida uma mensagem de sincronismo o dicionário de objetos é lido e são criadas as mensagens PDO. Depois da receção de PDOs a escrita do dicionário é atrasada até à receção de uma mensagem de sincronismo. Nesta altura todas as mensagens são processadas e os dados recebidos escritos no dicionário. Desta forma é possível definir com rigor os instantes de amostragem/atuação melhorando assim o controlo. Como a mensagem de sincronismo é recebida por todos os dispositivos ao mesmo tempo, faz com que a leitura e/ou escrita no dicionário seja realizada em todos os dispositivos ao mesmo tempo, melhorando assim aspectos como o *jitter*.

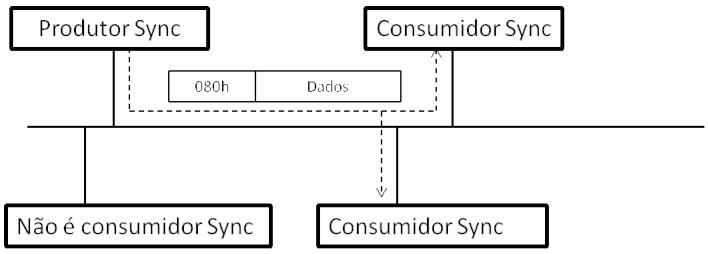

Figura 3.17: Esquema do serviço de sincronização (Adaptado a partir de [42]).

O objeto Sync é difundido periodicamente pelo produtor. O período de tempo entre mensagens Sync é definido pelo período cíclico de comunicação, que pode ser reconfigurado por uma ferramenta de configuração para os equipamentos.

A mensagem Sync é composta por uma simples mensagem CAN com um identificador predefinido e com o campo de dados vazio, pois a mensagem Sync transmitida ciclicamente apenas pretende indicar ao consumidor que pode iniciar um determinado comportamento. As PDOs síncronos usam a mensagem Sync como TT para a transmissão das mensagens.

Esta funcionalidade pode ser ligada no objeto Sync, com o índice 1005h. Além disso, esta entrada fornece o COB-ID da mensagem Sync.

Dispositivos implementados de acordo com CANopen CiA 301 Versão 4.1 e superior podem suportar uma mensagem Sync, que fornece um valor de contador Sync. Este valor do contador permite adaptar o comportamento síncrono de diferentes dispositivos na mesma rede.

O período de tempo entre as duas mensagens consecutivas Sync é chamado de "período do ciclo de comunicação"e pode ser ajustado no índice 1006h do dicionário de objetos (ver figura 3.18).

As mensagens PDO síncronas são transmitidas dentro de uma janela de tempo após a receção da mensagem Sync. Esta janela de tempo é designada por "janela de comprimento síncrona"e é configurável no objeto 1007h de todos os dispositivos, que têm de transmitir PDOs síncronos (ver figura 3.18).

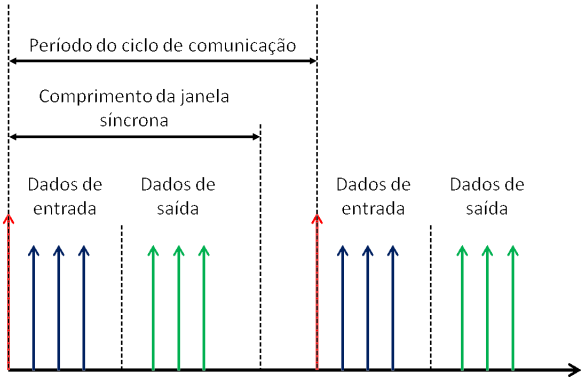

Figura 3.18: Esquema temporal da mensagem de sincronização (Adaptado a partir de [41]).

3.8.7 *Heartbeat*

O protocolo *Heartbeat* serve para o controlo de erro, como também pode assinala a presença de um nó e apresenta o seu estado. A mensagem *Heartbeat* é periódica de um nó para um ou vários outros nós. Ela indica que o nó transmissor ainda está trabalhar adequadamente (ver figura 3.19).

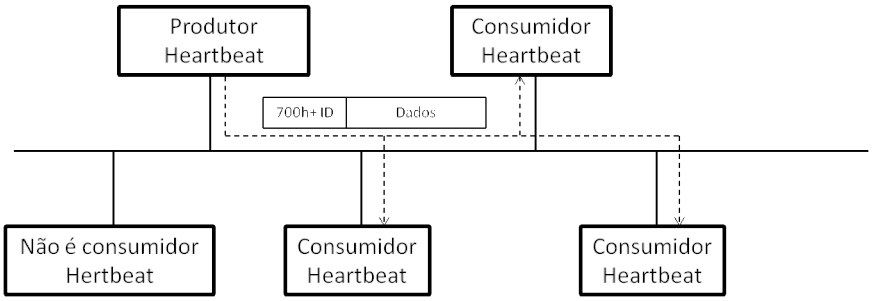

Figura 3.19: Esquema do serviço de comunicação *Heartbeat* (Adaptado a partir de [42]).

Além disso, no protocolo *Heartbeat* existe um serviço de controlo de erros antigos e não datados, o qual é chamado de *Node Protocol and Life Guarding*, que não é muito recomendado para ser implementado.

Um produtor de *Heartbeats* transmite uma mensagem cíclica, com o intervalo de tempo definido no objeto de comunicação apropriado. Um ou mais consumidores de *Heartbeats* podem receber esta informação. A relação entre o produtor e o consumidor é configurável através das entradas do dicionário de objetos. O consumidor de *Heartbeats* vigia a receção de mensagens dentro de um intervalo de tempo associado a aquele consumidor e se não for recebida uma mensagem durante um intervalo de tempo a definir ocorrerá um evento de erro.

3.8.8 Controlo de erro

A mensagem de emergência é causada por uma situação de erro interno do equipamento e é transmitida de um produto de emergência no equipamento de aplicação dedicado. Uma mensagem de emergência é transmitida apenas uma vez pelo evento de erro.

Como CANopen não é um sistema hierárquico mestre-escravo e o monitoramento do nó só transmite o estado da comunicação e não o estado atual do nó, cada nó requer identificador CAN de alta prioridade para indicar situações de erro. Tal mensagem de emergência é composta por oito bytes de dados da seguinte forma 3.20:

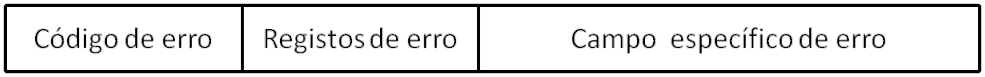

Figura 3.20: Estrutura da mensagem de erro.

O primeiro campo, designado por código de erro, é composto por dois bytes. A descrição destes códigos é organizada em categorias como se pode verificar na tabela A.12. O byte que se segue é descrito pela tabela A.13 e designado por registo de erro, é apresentada a causa da falha. Como a mensagem é composta por 8 bytes, os bytes seguintes é o campo em que o erro pode ser descrito pelo fabricante. Simultaneamente que é apresentado o código de erro e a causa de falha este são gravados no dicionário de objetos, nos índices 1002h e 1001h, respetivamente.

Os códigos de erro são especificados no DS 301. Simultaneamente com a transmissão da mensagem de emergência, o dispositivo grava o código de erro para o seu histórico de erros. O registo de erro é o conteúdo da entrada do objeto de dicionário com codificação *bit-wise* da causa do erro.

A reação do consumidor de emergência é específica da aplicação. CANopen define vários códigos de erro de emergência numa mensagem de emergência, que é basicamente uma mensagem CAN simples com 8 bytes de dados. O objeto de emergência permite que dispositivos indiquem erros de dispositivo internos. Quando receber este sinal, os outros participantes da rede podem avaliar as informações recebidas e iniciar ações apropriadas.

Capítulo 4

Implementação

4.1 Arquitetura da implementação

Nos níveis mais inferiores da hierarquia de fabrico existem várias soluções capazes de satisfazer distintas características que o ambiente possa ter. Estas redes atuam diretamente nos equipamentos que atuam fisicamente nos processos de fabrico de produtos. Cada vez mais os sensores e os atuadores têm capacidades de processamento e quando recorrem ao meio de comunicação devem de processar os dados de forma rápida. Os dados que são trocados não têm grandes *payloads*, por isso uma rede de campo deve ser sim capaz de cumprir requisitos temporais críticos.

A opção neste trabalho passou por usar a rede CAN para implementar a camada física e a camada de ligação de dados, um protocolo que se encontra vulgarmente na indústria automóvel e em sistemas de automação industrial, essencialmente devido à sua robustez. O CANopen foi o protocolo escolhido para a camada de alto nível. Este protocolo pode ser encontrado nos diversos sistemas médicos, industriais e domésticos. Além da solução adotada apresentar custos reduzidos, elevada robustez interferências eletromagnéticas e abertura do protocolo, mais características foram tidas em conta como:

- Baseia-se no conceito *broadcast*, *multicast* e *unicast*;
- Possui um estratégia de controlo de acesso ao meio que evita colisões sem ser destrutiva, baseando-se em priorização das mensagens;
- Um trama tem no máximo 8 bytes e todas são validadas por um campo de *checksum*;
- Realiza a retransmissão automática, caso o barramento esteja livre;
- Descreve os dispositivos, dados, parâmetro e funções, sob a forma de um dicionário de objetos, em que diferentes campos podem ser acedidos por um determinado índice e sub-índice;
- Todos os objetos podem ser acedidos através de um serviço baseado com o principio cliente/servidor;
- A transmissão de dados do processo de tempo crítico é baseada no principio do produtor/consumidor;
- Tem um padrão uniforme para a atribuição da prioridade das mensagens;
- Cada mensagem possuí um identificador que define a sua prioridade no barramento;

A rede escolhida para fazer ligação entre os módulo I/O garante também escalabilidade. Isto é, este sistema pretende ser robusto e fiável mesmo que o número de nós aumente, caraterística esta importante num sistema distribuído (ver figura 4.1).

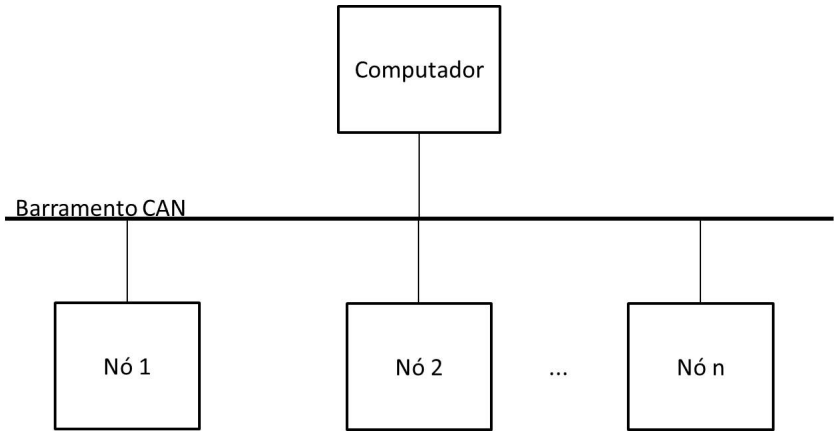

Figura 4.1: Arquitetura global da solução para o sistema pretendido.

No intuito de ser desenvolvido um sistema distribuído com o protocolo CANopen foram implementados dois nós recorrendo essencialmente ao seguinte material:

- Microcontrolador 32 bits da Microchip (PIC32MX795F512L);
- Compilador XC32;
- Placa de desenvolvimento Explorer 16;
- PICtail CAN/LIN/J2602;
- Pilha protocolar CANopenNode;

4.2 Módulo I/O - CANopen

O módulo inclui o *firmware* pré-programado para a comunicação CANopen para entradas e saídas, digitais e analógicas, com rotinas de diagnóstico implementadas pelo protocolo. Todas as entradas e saídas, bem como os parâmetros de configuração são acessíveis através do protocolo CANopen, porque este dispositivo está de acordo com o perfil de dispositivo CiA DS 401 "*Generic I/O Devices*"e o perfil de comunicação CiA DS 301. Basicamente o perfil de dispositivo CiA DS 401 possibilita a interação, no que toca à troca de dados relativa às entradas e saídas digitais e analógicas, com outros dispositivos que implementem o mesmo perfil de dispositivo (ver figura 4.2).

Para testar a pilha protocolar implementada e para estudar o protocolo CANopen, foram usados dois nós com os respetivos módulos:

- Três entradas digitais;
- Três saídas digitais;
- Uma entrada analógica;
- Uma saídas analógica;
- *Liquid Crystal Display* (LCD);

- *Electrically Erasable Programmable Read-Only Memory* (EEPROM);
- Dois *Rotary switchs*;
- Um potenciómetro;
- Um *Light Emitting Diode* (LED).

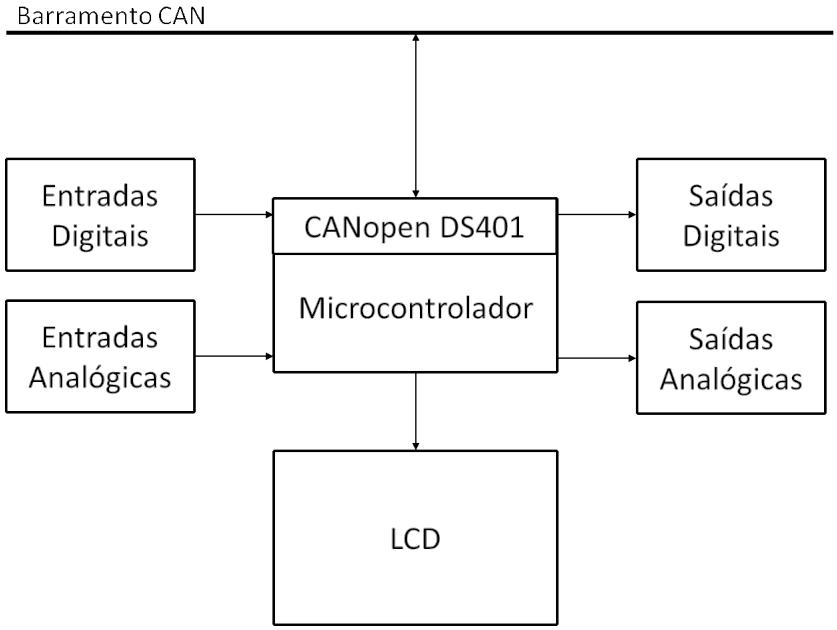

Figura 4.2: Componentes principais de um nó.

4.2.1 Camada física

Apesar de haver implementações CANopen que usam diferentes suportes físicos, todos os suportes devem ser especificados de acordo com a ISO 11898. O sinal diferencial usado pela ISO 11898 oferece bons níveis de proteção para os ruídos eletromagnéticos, muitas vezes encontrados no ambiente industrial.

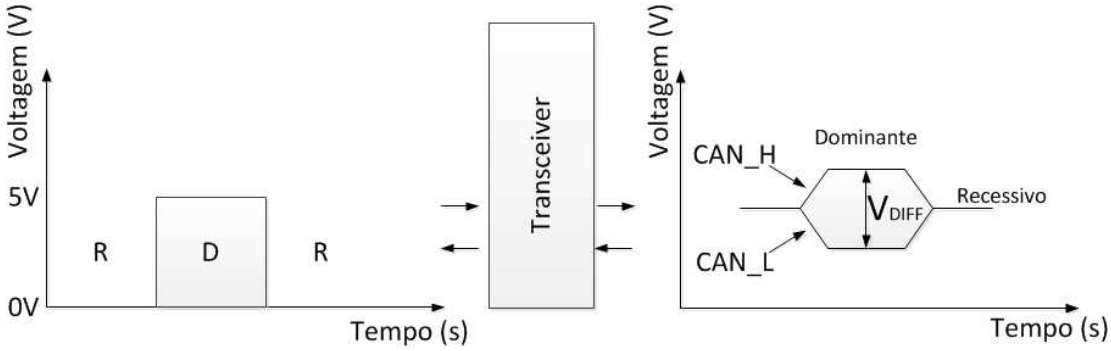

Figura 4.3: *Transceiver* de Alta Velocidade Compatível.

Os *transceivers* estão conectados entre o controlador CAN e o meio físico de transmissão. No meio industrial muitas vezes é utilizado o par de fios, de preferência com fios

torcidos opcionais para os sinais adicionais personalizados ou blindagem (ver figura 4.3). A figura 4.3, ilustra a função do *tranceiver*, que passa pela conversão do sinal *Transistor-Transistor Logic* (TTL) vindo do pino de transmissão do controlador CAN num sinal diferencial entre os dois fios, tipicamente designados por CAN_H e CAN_L, (ver figura 4.3).

Como a aplicação foi realizada tendo por base o microcontrolador na placa de desenvolvimento Explorer 16, a PICtail CAN/LIN/J2602 foi a melhor solução encontrada para implementar a interface CAN. A PICtail CAN/LIN/J260 conecta dois *transceivers* de comunicação CAN com os módulos CAN do microcontrolador instalado na placa de desenvolvimento. O *transceiver* utilizado por esta placa é o MCP2551 da Microchip, é um controlador de alta velocidade, tolerante a falhas, que serve como interface entre o controlador CAN e o barramento físico. Este componente é completamente compatível com a norma ISO 11898 e permite a implementação de redes de alta velocidade [49].

Número do pino	Nome do pino	Descrição
1	TXD	Entrada de dados a transmitir
2	Vss	GND
3	Vdd	Tensão de alimentação
4	Rxd	Saída de dados recebidos
5	Vref	Tensão de referência
6	CAN_L	CAN-Low
7	CAN_H	CAN-High
8	RS	Slope-Control

Tabela 4.1: Descrição dos pinos do MCP2551.

Na tabela 4.1, são descritos os pinos que constituem o *transceiver* utilizado. É importante referir que o pino TXD é o responsável por transmitir os dados enviados pelo microcontrolador e o pino RXD é o responsável por enviar os dados para o microcontrolador.

Este *transceiver* permite selecionar três modos diferentes de operação [50]: *High-Speed*, *Slope-Control* e em modo *Standby*.

O modo *High-Speed* é selecionado através da ligação entre o pino RS e o pino VSS. Neste modo é possível suportar a velocidade máxima.

O modo *Slope-Control* é selecionado por intervenção das resistências ligadas entre o pino RS e o pino VSS. Neste modo a velocidade de transição de nível do CAN_H e do CAN_L é limitada, de modo diminuir possíveis interferências eletromagnéticas. O declive da transição é proporcional ao valor da corrente que passa pela resistência.

O terceiro modo, o modo *Standby*, é realizado através da ligação dos pinos RS ao VDD, não havendo lugar a qualquer comunicação.

No presente trabalho o *transceiver* permanece constantemente no modo *High-Speed*, uma vez que se pretende testar o sistema com a taxa de transmissão máxima e porque de momento não está prevista grande instabilidade provocada pelas interferências eletromagnéticas.

Uma vez que o protocolo CANopen é utilizado em distintas aplicações, o cabeamento e os conetores não fazem parte da especificação. Mas existe uma publicação em [51], designada por CiA *Draft Recomendation Proposal* 303-1, que apresenta recomendações sobre os pinos para vários tipos de conetores.

O conector utilizado no trabalho foi o 9-Pin D-Sub que apresenta o seguinte alinhamento e pinos utilizados, ilustrado na figura 4.4 e na tabela 4.2.

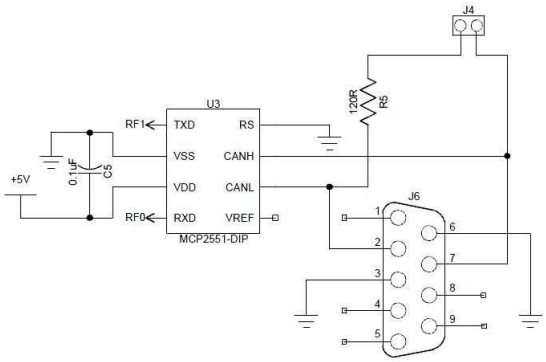

Figura 4.4: Esquema elétrico do *transceiver* [49].

Pino	Sinal	Descrição
2	CAN_L	Canal CAN_L (nível mais baixo)
3	CAN_GND	Canal terra CAN
6	(GND)	Canal terra (opcional)
7	CAN_H	Canal CAN_H (nível mais alto)

Tabela 4.2: Alinhamento dos pinos no conector 9-pin D-Sub.

A disposição física da rede CANopen apresentada é linear. A linha principal consiste nos sinais CAN_L e CAN_H com resistências de terminação. É recomendado o uso das resistências de 120 Ohm para taxas de transmissão de 1 Mbps e para taxas de transmissão mais baixas o uso de resistências de 150 Ohm a 300 Ohm. Na PICtail já se encontram montadas resistências de terminação de 120 Ohm.

4.2.2 Pilha Protocolar CANopen

Os requisitos de processamento do protocolo CANopen variam com as funcionalidades implementadas para a comunicação. Apesar deste protocolo ser flexível quanto a algumas funcionalidades implementadas, por vezes é necessário alguns requisitos temporais mínimos, que podem ser na ordem da fração de milésimo de segundo, definido com um tempo de reposta garantido a um evento e que impliquem velocidades de transmissão mais elevadas. Por exemplo, uma aplicação para controlo de sensores de temperatura, tais requisitos podem não ser necessários, ou seja um tempo de resposta mais prolongado a uma variação de temperatura não tenha consequências. Mas, por exemplo, num controlo de movimento já poderia se este controlo exigisse grandes precisões.

Do levantamento efetuado havia duas pilhas protocolares disponíveis para implementar o protocolo CANopen com um custo reduzido.

A biblioteca MicroCANopen não necessita de elevados recursos tornando-a adequada para o uso de microcontroladores de baixos recursos como, por exemplo, os de 8 bits. Esta biblioteca implementa apenas um conjunto mínimo de funcionalidades do protocolo

de forma a ser usada nos referidos microcontroladores. Como está escrita em linguagem C permite que seja introduzida ou removidas funcionalidades.

Os requisitos do espaço de memória do código para a implementação da pilha protocolar do protocolo também podem variar. Não só a compilação do código C pode ser diferente para várias arquiteturas de microcontroladores como o tamanho do código pode variar com as funcionalidades implementadas.

Os requisitos do espaço da memória de dados dependerão de fatores semelhantes aos dos requisitos de memória de código, podem rondar entre os 100 e 200 bytes para as implementações mínimas e cerca de 1 kbytes para implementações mais completas. Os nós que precisem monitorizar todas as variáveis do processo podem precisar de várias centenas de bytes de RAM adicional [41].

A pilha protocolar CANopenNode também é desenvolvida em código C. Esta pilha protocolar começou por ter enfoque nos microcontroladores de 8 bits e com o passar do tempo o enfoque passou para os microcontroladores de 32 bits, devido à necessidade crescente de tempos de resposta ou de atuação.

Como foi anteriormente referido, a pilha protocolar apresenta várias versões, facilitando por isso a implementação do protocolo numa gama mais ampla de microcontroladores. A versão utilizada neste trabalho foi a última versão desta pilha protocolar padronizada, pela norma EN50325, como protocolo de alto nível, pode ser implementada vários microcontroladores como:

- dsPIC30F;
- PIC24F;
- dsPIC33F;
- PIC32;

Tal como na pilha protocolar MicroCANopen, a pilha protocolar CANopenNode também permite que possam ser acrescentadas ou removidas funcionalidades. Mas o MicroCANopen é uma pilha protocolar dirigida aos microcontroladores de 8 bits, enquanto a pilha protocolar CANopenNode, apresenta várias versões, nas quais as mais antigas tinham o foco nos microcontroladores de 8 bits e as mais recentes no microcontrolador de 32 bits da Microchip. A possibilidade de ter várias versões disponíveis da pilha protocolar permite que principalmente o código seja dedicado aos recursos de um microcontrolador, beneficiando por isso em aspetos como o espaço de memória ocupado pelo código e pelo comportamento em tempo-real do microcontrolador.

Na tabela 4.3, é realizado um sumário das funcionalidades das duas pilhas protocolares mencionadas e na primeira coluna é apresentado o que o protocolo CANopen oferece nas suas especificações, relativamente a cada particularidade mencionada na pilhas protocolares. Como se pode verificar através desta tabela, a pilha protocolar que apresenta uma implementação mais completa do protocolo CANopen é a ultima versão da pilha protocolar CANopenNode. As principais funcionalidades que a ultima versão do CANopenNode tem a mais que o MicroCANopen é a configuração dinâmica do serviço PDO e o serviço de consumo das mensagens *Heartbeat*. A configuração dinâmica do serviço PDO significa que será possível fazer a configuração deste serviço no estado operacional, por outras palavras, não é necessário parar o sistema para fazer as respetivas configurações. O serviço de consumo das mensagens *Heartbeat* possibilita que haja nós dependentes do funcionamento de outros.

	CANopen	MicroCANopen	CANopenNode
Bitrates CAN (kbps)	10, 20, 50, 125, 250, 500, 800, 1000	10, 20, 50, 125, 250, 500, 800, 1000	10, 20, 50, 125, 250, 500, 800, 1000
Máx. nós por segmento	127	127	127 (55)
Network management	Originalmente projetado para usar um mestre, mas pode operar sem ele	Originalmente projetado para usar um mestre, mas pode operar sem ele	Originalmente projetado para usar um mestre, mas pode operar sem ele
Node guarding/ Heartbeat	Node Guarding realizado pelo mestre da rede supervisão Heartbeat realizada por qualquer nó	Todos os nós produzem o sinal Heartbeat, para puderem ser monitorados	Todos os nós podem produzir um sinal Heartbeat, para serem monitorizados e para monitorar outros nós.
Configuração dos nós	Nós tipicamente configurados via CAN	Nós são pré-configurados, podem ser configurados via CAN mas as configurações não podem ser realizadas durante a operação	Nós são pré-configurados, podem ser configurados via CAN e as configurações podem ser realizadas durante o modo operacional
Dicionário de objetos: Variáveis de processo de dados	Disponível (8 ou 16 bit)	Indisponível no dicionário de objetos	Disponível (8 ou 16 bit)
Dicionário de objetos: Configuração de dados de processo	Disponível	Indisponível	Disponível
Dicionário de objetos: Suporta longas variáveis	Suporta variáveis e campos de dados com qualquer comprimento de dados	Uma entrada do dicionário de objetos não pode ter mais que 4 bytes	Uma entrada do dicionário de objetos não pode ter mais que 4 bytes
Mapeamento de múltiplas variáveis numa mensagem CAN	Suporte dinâmico para re-mapeamento de variáveis em mensagens CAN	Suporta uma configuração	Suporte dinâmico para re-mapeamento de variáveis em mensagens CAN
Métodos de acionamento de mensagens com dados de processos	Qualquer combinação: mudança de estado, de síncrona ou específica do fabricante; Tempo de inibição	Suporta a mudança de estado, baseado em tempo, e suporta tempo de inibição	Qualquer combinação: mudança de estado, de síncrona ou específica do fabricante; Tempo de inibição

Tabela 4.3: Tabela comparativa das pilhas protocolares abordadas.

4.2.3 Hardware

Nas figuras 4.5 e C.2 estão representadas o diagrama global do módulo I/O implementado e estão representados a ligação entre os blocos mais elementares e a eletrónica necessária para a implementação.

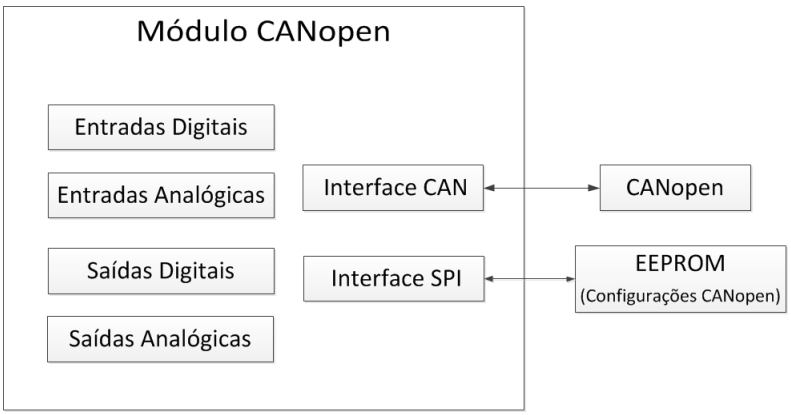

Figura 4.5: Esquema da arquitetura do módulo I/O.

O *hardware* essencial para a implementação de um nó, para comunicar num barramento CAN, é apresentado pela figura 4.6. Para que este nó se aproximasse mais de um módulo I/O e pudesse ser realizadas algumas simulações foram adicionados mais alguns componentes que irão ser apresentados mais à frente. Antes destes serem apresentados irá ser justificada a escolha do microcontrolador utilizado e depois serão apresentados os outros componentes utilizados para a implementação do nó CANopen.

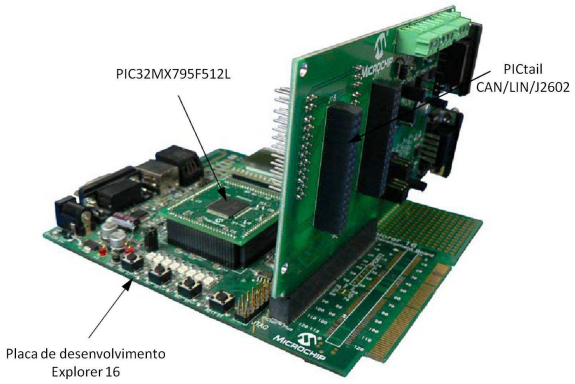

Figura 4.6: Hardware utilizado para o nó.

PIC32MX795F512L

A família dos microcontroladores de 32 bits, o PIC32, faz parte dos produtos com melhores recursos na Microchip. Oferece melhorias significativas na rapidez, memória e desempenho comparando com as outras famílias de 16 e 8 bits. As ferramentas de desenvolvimento como o compilador C da Microchip são similares aos usados na família 16 bits.

Para uma breve comparação de famílias, na tabela 4.4, são comparados alguns atributos relevantes com três microcontroladores, o PIC32MX795F512L, o PIC24FJ64GA002 e o PIC16F887.

PIC32	**PIC24F**	**PIC16**
32 bit	16 bit	8 bit
512 KB Flash	64 KB Flash	8 KB Flash
128 KB RAM	8 KB RAM	386 B RAM
80 MHz	32 MHz	20 MHz
80 MIPS	16 MIPS	5 MIPS
3.3 V	3.3 V	5 V

Tabela 4.4: Comparação entre as três famílias do microcontroladores da Microchip.

Por exemplo, como é mostrado na tabela 4.4, o microcontrolador de 32 bits é cinco e dezasseis vezes mais rápido, que o de 16 e 8 bits respetivamente, para além de executar instruções com dados maiores. A figura 4.7 apresenta o diagrama interno do microcontrolador escolhido. O PIC32 é organizado internamente por dois barramentos. O barramento superior corre à taxa do relógio do sistema do CPU, fazendo deste o barramento mais rápido [52]. O barramento mantém os seguintes componentes conetados:

- DMA (*Direct Memory Access*);
  - Oito canais com deteção automática do tamanho de dados;
  - Controlo de redundância de 32 bits;
  - São adicionados seis canais dedicados ao CAN, USB e Ethernet;
- USB;
- Dois canais CAN;
- Interface Ethernet;
- Flash;
- Memória RAM;
- Controlador das interrupções;
- Portas digitais;
- Uma ponte periférica;

É de relevar o aspecto que as portas digitais residem no barramento de alta velocidade. Isso significa que eles pode ser alternar o seu valor lógico à taxa de 80 MHz, capaz de gerar sinais digitais de até 40MHz, se necessário. Com a utilização da DMA, pode-se aceder diretamente à memória, sem dependência da intervenção *Central Processing Unit* (CPU) para a transferência de dados.

O outro barramento é denominado por barramento periférico. Neste barramento são conetados: um RTCC (*Real Time Clock Calendar*), seis UART (*Universal Asynchronous Receiver Transmitter*), três SPI (*Serial Peripheral Interfaces*), quatro I2C (*Inter-Integrated Circuits*), dezasseis canais ADC (*Analog-to-Digital Converter*) de 10 bits, uma PMP (*Parallel Master Port*), cinco CCP (*Compare/Compare Ports*), e dois comparadores analógicos.

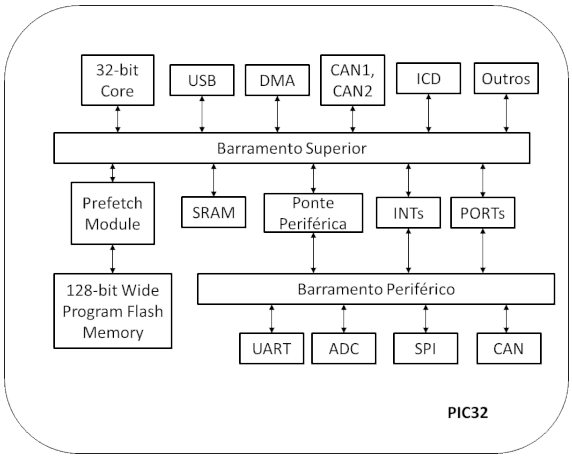

Figura 4.7: Diagrama da arquitetura geral do PIC32.

Requisitos do microcontrolador

Um dos principais requisitos para a elaboração do trabalho passou pela realização de um estudo sobre qual a categoria que se insere a implementação da rede CANopen, porque em geral para aplicações de baixo volume tendem a ser menos sensíveis ao preço, podendo desta forma sobre dimensionar os requisitos do microprocessador ou microcontrolador utilizado, e por consequência facilitar a implementação CANopen. No entanto para aplicações de grande volume é importante delimitar os recursos requisitados para disponibilizar a execução do protocolo de comunicação CANopen, por causa dos custos que poderão advir.

O protocolo CANopen é bastante flexível, por isso os requisitos de processamento também vão variar mediante as funcionalidades implementadas. Como anteriormente foi referido, até pode ser implementado num microcontrolador de 8 bits executando velocidades que não comprometam o bom funcionamento. Mas é necessário ter em mente o compromisso entre a velocidade e a taxa de ocupação do barramento CAN, necessidades de processamento local do nó e os requisitos temporais de resposta, quando chega o momento de selecionar o microcontrolador a usar.

Outro aspecto a ser tido em conta é o espaço de memória do código e de memória dos dados. O espaço de memória do código para a implementação da pilha protocolar CANopen pode variar muito, pela razão anteriormente referida da flexibilidade e modularidade do protocolo.

Em cenários mais exigentes a taxa de transmissão utilizada é de 1 Mbps e a mensagem enviada poderá não ter campo de dados. Por isso, pode-se analisar os requisitos básicos exigidos refletindo na mensagem mais pequena, de 0 bytes de dados, que corresponde a 50 tempos de bit no barramento e que por sua vez corresponde a 50 microssegundos a uma taxa de transmissão de 1 Mbps. Desta forma, o microcontrolador a escolher deve ser capaz de tratar cada mensagem em menos de 50 microssegundos.

Ao longo dos tempos diferentes controladores foram introduzidos nas implementações CAN. Ao longo desta evolução os controladores para este tipo de implementações com a melhor abordagem são aqueles combinam o Full CAN com *First In, First Out* (FIFO) dedicados para cada mensagem. Apesar desta ser a melhor abordagem pode ser a mais complexa para configurar, especialmente quando cada FIFO é dedicada a cada mensa-

gem. No caso do PIC32 possui até 32 mensagens FIFO, em que cada FIFO pode ter 32 mensagens, que perfaz o total de 1024 mensagens possíveis de serem tratadas guardadas em memória *Random Access Memory* (RAM).

A escolha do microcontrolador PIC32MX795F512L e da pilha protocolar CANopenNode deve-se essencialmente ao facto da pilha protocolar disponibilizar uma versão mais completa do protocolo CANopen, pelos recursos do microcontrolador que permitem maior facilidade para interagir com outros dispositivos com o mesmo protocolo e por não se comprometer comportamento em tempo-real.

Entradas e Saídas Digitais

As estradas digitais possuem dois estados lógicos possíveis, zero ou um. Na indústria estes entradas podem estar ligados a vários sensores como:

- Fins de curso;
- Sensores de proximidade indutivos ou capacitivos;
- Comutadores;
- Pressostatos;
- Controlo de nível;

As saídas digitais fazem parte da família dos atuadores como:

- Relês;
- Contadores;
- Solenoides;
- Válvulas;

Para efeitos de simulação foram utilizadas três entradas digitais e três saídas digitas. O microcontrolador PIC32MX795F512L dispõe num total de 81 linhas de I/O de 1 bit, organizados em 7 portos, denominados de PORTA, PORTB, PORTC, PORTD, PORTE, PORTF E PORTG. O PORTA agrupa 12 linhas de 1 bit configuráveis, como entrada ou como saída, identificadas pelas siglas RA0, RA1, RA2, RA3, RA4, RA5, RA6, RA7, RA9, RA10, RA14, RA15. Cada um dos outros portos designará as suas linhas configuráveis como entrada ou como saída identificadas da mesma maneira como no porto A.

Na placa de desenvolvimento Explorer 16 estão ligados, aos pinos RA0, RA1, RA2, RA3, RA4, RA5, RA6 e RA7, sete LEDs para atuarem como saídas digitais e os pinos RA7, RD6, RD7 e RD13 estão ligados a quatro botões de pressão para atuarem como entradas digitais.

A informação sobre o estado das entradas e saídas do microcontrolador também são armazenadas no dicionário de objetos, de acordo com o perfil de dispositivos CiA DS 401, para que sempre que, o microcontrolador execute um ciclo de leitura das entradas ou execute uma modificação nas saídas, leia ou escreva no dicionário de objetos para que a informação esteja disponível para ser utilizada pelo meio de comunicação. Os valores provenientes das três entradas digitais estão ligadas ao objeto de índice 6000h e sub-índice 01h e, das três saídas digitais estão ligadas ao objeto de índice 6200h e sub-índice 01h, conforme define o perfil de dispositivos CiA DS 401.

Entrada analógica

Na indústria podemos encontrar entradas analógicas como um sensor de pressão, de nível ou de corrente. No processamento de sinais analógicos, a conversão para sinais digitais é fundamental, para que possam ser processados por um microcontrolador. Os elementos que efetuam a conversão de um sinal analógico num digital são designados por conversores analógico-digital ou *Analog-to-Digital Converter* (ADC).

No presente trabalho é usado uma ADC de 10 bits. Ter uma ADC de 10 bits significa que o resultado da conversão vem representado com uma resolução de $\frac{1}{1024}$ gama máxima. Por exemplo, é possível distinguir valores com cerca de 5mV de resolução aproximadamente ($\frac{5}{1024} = 0,004883$V), numa gama máxima de valores a medir de 5V, ou seja uma gama entre 0 e 5V.

Através de potenciómetro, ligado ao pino AN2, é possível utilizar uma entrada analógica, que vai ser lida através do *buffer* ADC1BUF0. Este valor é acedido pelo protocolo CANopen no objeto de índice 6401h e sub-índice de 01h, conforme o o perfil de dispositivos CiA DS 401.

Pulse Wide Modulation

São várias as aplicações onde há necessidade de utilizar a *Pulse Width Modulation* (PWM), como por exemplo, no controlo de velocidade de motores de corrente contínua, na área da automação e da robótica. Ao contrário dos motores de passo e dos motores de corrente alternada, o controlo da velocidade de um motor DC é relativamente simples. A técnica consiste em variar a tensão média aplicada ao motor, mais especificamente na armadura do motor. A tensão média é regulada pela técnica de *duty cycle*, que consiste na determinação do tempo, na onda quadrada, em que está a ser aplicada tensão. O PWM é por isso uma técnica que permite controlar a tensão média aplicada, por exemplo, a um motor, variando desta forma a velocidade de rotação. Um motor DC é dividido em 6 partes principais:

1. Estator;
2. Rotor;
3. Armadura;
4. Coletor;
5. Escovas;
6. Terminais.

Um sinal elétrico PWM observado num osciloscópio tem a forma de uma onda quadrada, como se pode ver na figura 4.8, adquirida experimentalmente a partir do programa EasyScope. Esta onda quadrada consiste numa onda em que tem uma tensão nula, durante um tempo t_1, e uma tensão diferente de zero, durante um tempo t_2. Através de uma saída digital do microcontrolador é possível gerar uma onda quadrada, que durante um tempo t_2 valha a tensão de alimentação, designada por VDD.

A maioria dos microcontroladores PIC32 apesar de ter entradas e saídas digitais e entradas analógicas, não têm saídas analógicas. Por isso foi utilizado um pino com a possibilidade de processar um sinal em PWM do PIC32MX795F512L para implementar uma saída analógica. Neste microcontrolador a PWM é uma função do módulo *Output*

Compare. Para que experimentalmente se tivesse perceção física da mudança do *duty cycle*, usou-se um LED para ver a variação da intensidade de luz de um LED ao mudar o *duty cycle*.

O valor da saída PWM está ligado a um objeto fornecido pelo perfil de dispositivo implementado, designado por saída analógica de 16 bits, que é acedida através do índice 6411h e sub-índice 01h, uma vez que era o objeto deste perfil de dispositivo que mais se adequava à ligação desta variável.

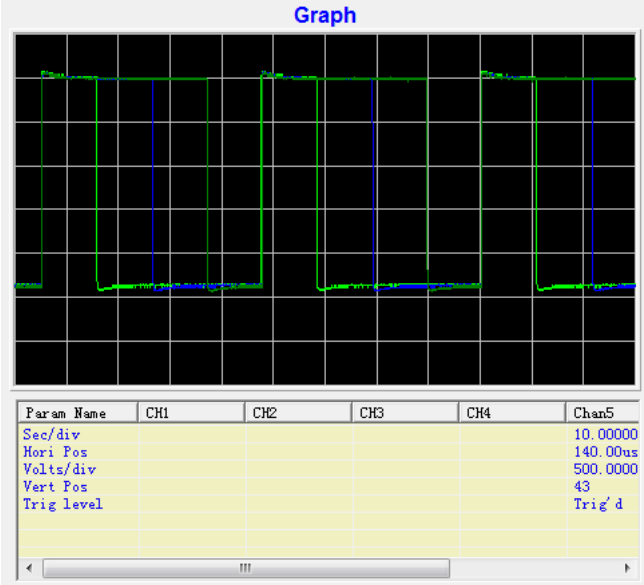

Figura 4.8: Gráfico obtido experimentalmente de duas PWMs sobrepostas.

EEPROM

Uma EEPROM é um tipo de memória utilizado para armazenar os dados provenientes do dicionário de objetos do dispositivo. Ao contrário de uma memória programável apagável somente de leitura (EPROM), na EEPROM pode ser programada e apagada várias vezes, eletricamente. Pode ser lida um número ilimitado de vezes, mas só pode ser apagada e programada um número limitados de vezes, que varia entre cem mil e um milhão de vezes. Esse limite é causado pela contínua deterioração interna do *chip* durante o processo de apagar, porque requer um tensão elétrica mais elevada.

O integrado usado, 25LC256, é uma memória EEPROM de 256 Kbits, ou seja 32768 bytes organizados em páginas de 64 bytes cada. Cada posição de memória tem 8 bits de dados. O microcontrolador pode ler ou escrever nesta memória através de uma interface *Serial Peripheral Interface* (SPI). Esta memória pode ser alimentada com tensão compreendida entre 2,5 até 5,5 Volt.

A EEPROM é utilizada nesta implementação para que todas as vaiáveis do dicionário de objetos, que estejam gravadas na memória *Read Only Memory* (ROM), possam ser gravadas e acedidas. Por outras palavras, a faculdade de puder gravar estas variáveis neste tipo de memória garante que as configurações que se queira que permaneçam, não sejam apagadas por causa de uma reinicialização do dispositivo. Esta função já é implementada através da pilha protocolar utilizada, mas foram realizadas algumas

modificações que passaram fundamentalmente pela adaptação de um diferente integrado usado, isto é, com características diferentes como o tamanho da memória utilizada. A função da implementação da EEPROM é opcional, isto é, pode ser desativada através do *firmware*, ou por outras palavras o código para ativar esta função está dependente de um #define.

Rotary Switchs

Neste trabalho foram implementados dois *rotary switchs* para codificar o ID e a taxa de transmissão utilizada pelo dispositivo (ver figura 4.9). No que toca à configuração do ID as diferentes posições do *rotary switch* poderão dar quinze valores diferentes. Quanto à taxa de transmissão, a tabela 4.5 apresenta as transmissões possíveis nas sete diferentes posições.

Posição	Taxa de transmissão (kbit/s)
0	10
1	50
2	125
3	250
4	500
5	800
6	1000

Tabela 4.5: Taxas de transmissão das respetivas posições do *rotary switch*

A necessidade de ter dois *rotary switchs* passava por dar ao utilizador a possibilidade de configurar as duas variáveis importantes para ligar o nó. Desta forma, os valores traduzidos pela posição dos *rotary switchs* seriam assumidos na inicialização do dispositivo e se fosse o caso de estes valores serem mudados, o nó teria ser reinicializado. Esta reinicialização poderia ser feita através do corte de energia ou através do serviço NMT fornecido pelo protocolo CANopen.

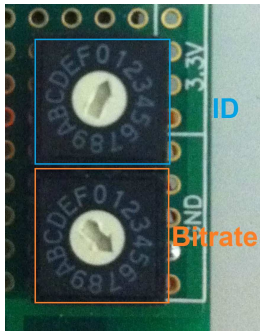

Figura 4.9: Fotografia dos *rotary switchs*.

LCD

Um simples LCD torna o projeto muito mais amigável ao utilizador além de aumentar a gama de funções e operações com um custo relativamente baixo. O facto da simplicidade de como se configura um LCD para trabalhar em conjunto com um microcontrolador e pelo facto de que este já vem implementado na placa de desenvolvimento utilizada fez com que se tivesse implementado.

O LCD usado é de 2 linhas de 16 caracteres sem retro iluminação e monocromático. Desta forma, o LCD mostra a taxa de transmissão selecionada no dispositivo, e de forma percentual o estado da entrada e da saída analógica.

O LCD 16 pinos de acesso:

- 8 pinos de dados;
- 3 pinos de controlo;
- 3 pinos de alimentação.

Como pode ser visto na figura 4.10, o LCD apresenta a taxa de transmissão configurada no nó e o valor em percentagem da entrada analógica e do *duty cycle*. Tal como na implementação da EEPROM esta pode ser desativada através da reprogramação do *firmware* do microcontrolador.

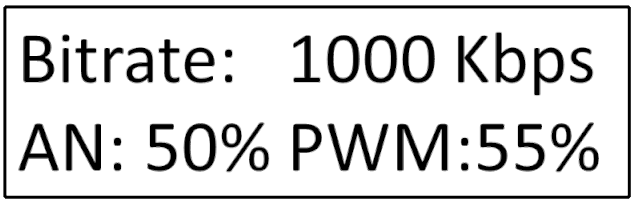

Figura 4.10: Ilustração da informação apresentada pelo LCD.

4.2.4 Outros componentes

Dos sete LEDs que a placa de desenvolvimento Explorer16 disponibiliza, até aqui só foi apresentado a funcionalidade de três para simular saídas digitais. Os outros quatro que a placa de desenvolvimento oferece, tem o papel de dar o *feedback* do estado de operação do nó. Como é apresentado na tabela 4.6, os LEDs que estão ligados aos pinos RA2, RA3 e RA4 dão o *feedback* sobre serviços disponibilizados pelo protocolo CANopen. Os respetivos LEDs estarão ligados se o dispositivo estiver no estado operacional do serviço NMT, e *Heartbeat*, se estiver a produzir e a consumir um sinal *Heartbeat*. Os LEDs que estão ligados aos pinos RA0 e RA1 dão o *feedback* sobre o estado do nó, de acordo com a especificação CiA *Draft Recommendation* (DR) 303-3 [40].

Ao contrário dos LEDs ligados aos pinos RA2, RA3 e RA4 os pinos RA0 e RA1 dão o *feedback* sobre todo o programa que corre no microcontrolador (ver tabela 4.6).

Através LED ligado ao pino RA1, como especifica CiA DR 303-3, sinalizará os seguintes estados:

- desligado do barramento, se estiver sempre ligado;
- erro na receção da mensagem SYNC, se fizer a sequência de 200 ms *on*, 200 ms *off*, 200 ms *on*, 200 ms *off*, 200 ms *on*, 1 s *off*;
- erro no serviço *Heartbeat*, se fizer a sequência de 200 ms *on*, 200 ms *off*, 200 ms *on*, 1 s *off*;
- aviso de um *warning* se piscar num intervalo de 100 ms;
- ligado se estiver num estado de erro geral.

O LED ligado ao pino RA0, como especifica o CiA DR 303-3, sinaliza três estados NMT, os estados:

- operacional, em que estará sempre ligado;
- pré-operacional, se piscar em intervalos de 100 ms;
- parado, se este estiver ligado durante 200 ms e desligado durante 1 s.

Pino	Descrição
RA0	Operacionalidade da comunicação CAN
RA1	Estado de erro do nó
RA2	Estado NMT operacional
RA3	Serviço *Heartbeat consumer*
RA4	Serviço *Heartbeat producer*

Tabela 4.6: LEDs com tarefas de *feedback* sobre a comunicação.

4.3 Interface gráfica de configuração e análise

Um teste de conformidade não testa apenas a performance física do dispositivo. Neste teste é importante confirmar se a EDS apresenta os parâmetro de comunicação aplicadas e se as entradas do dicionário de objetos estão em conformidade com os requisitos mínimos para estar conforme com o protocolo CANopen.

Há uma variedade de maneiras para se realizarem os testes de conformidade dos dispositivos CANopen. Em primeira instância foi possível adquirir um conversor *Universal Serial Bus* (USB)/CAN (ver figura 4.11). Através do dispositivo CAN BUS Analyzer da Microchip, foi possível ter uma interface para que se pudesse dar ao computador acesso ao barramento CAN, mas este dispositivo apenas permitia enviar mensagens de teste para o dispositivo e analisar as suas respostas. Este caminho exigia um profundo conhecimento sobre o sistema de mensagens CANopen porque este dispositivo não disponibilizava a *Dynamic Link Library* (DLL), para que fosse desenvolvida uma interface, que apresentasse funcionalidades mais interessantes para configurar os nós. Por outras palavras, o primeiro objetivo passava por criar uma interface que não fosse preciso ter o conhecimento sobre a estrutura das mensagens CANopen e o segundo objetivo configurar os nós sem que fosse necessário grande conhecimento sobre o protocolo CANopen.

Neste trabalho foi por isso realizada uma pesquisa, para que se pudesse fazer um levantamento do que o mercado oferecia no âmbito dos conversores USB/CAN e sobre os *softwares* de configuração.

No que toca aos conversores USB/CAN destaco alguns dispositivos encontrados e respetivos fabricantes, que se adequavam ao desenvolvimento de uma interface de configuração:

- PEAK System (PCAN-Light API) [53];
- Carambola (USB2CAN) [54];
- MHS Elektronik GmbH (Tiny-CAN API) [55];
- ESD (CAN USB 2.0 Interface) [56];
- IXXAT (USB-to-CAN compact) [57];
- Zanthic Technologies CAN4USB FX [58];
- Lawicel CANUSB [59];
- KVASER CANLIB [60];
- EMS Dr. Thomas Wünsche [61];

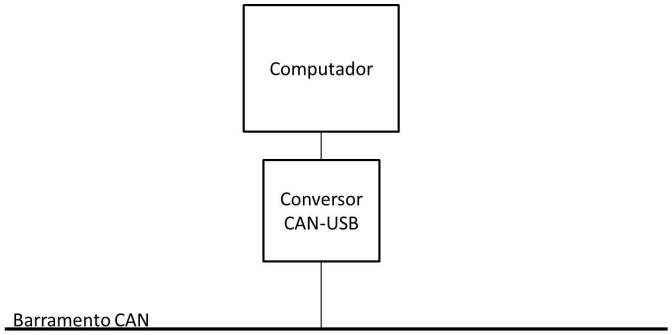

Figura 4.11: Interface entre o computador e o barramento.

A possibilidade de ter acesso aos parâmetros e às funções pela rede de comunicação, torna-se essencial no que toca a novas configurações sem reprogramação do nó. Alguns *softwares* encontrados apresentavam licenças oficiais que acarretavam elevados custos e por isso mesmo optou-se por começar um caminho de desenvolvimento de uma interface onde se pudessem realizar, principalmente, configurações e análises dos dispositivos.

Na fase de projeção da arquitetura da interface, foi realizado um levantamento das funcionalidades básicas e do que o mercado já oferecia. É de realçar também um aspeto importante, que ao terem sido pesquisados outros *softwares*, foi mais fácil projetar uma interface que aproveitasse melhor a flexibilidade de configuração que o protocolo oferece.

As funcionalidades básicas que a interface deve ter, passam pelo acesso ao tráfego do barramento, a possibilidade de ver e escrever tramas a partir do computador pessoal e ter os endereços de todos os objetos implementados no nós.

4.3.1 Conversor USB-CAN

Neste trabalho recorreu-se a um conversor USB-CAN, produzido pela Peak-System, para interligar o computador ao sistema, uma vez que era um dos dispositivos, anteriormente mencionados, que disponibilizava a DLL para desenvolver uma interface que permitisse explorar melhor as funcionalidades fornecidas pelo protocolo CANopen, que podem ser configuradas via CAN desde que os nós estejam ligados ao barramento comum. Este

conversor tem por base um controlador SMA1000 da NXP e no transdutor PCA82C251 da NXP [53]. Com o conversor é disponibilizado documentação, exemplos, bibliotecas, que funcionam tanto em sistemas operativos da Windows como Linux, e uma aplicação de monitorização do barramento que permite a visualização de mensagens CAN.

Na figura 4.12, apresenta uma fotografia do sistema desenvolvido. Como anteriormente foi referido, este consiste em dois nós, representados a verde, num conversor USB/CAN, representado a vermelho e pela interface de configuração dos nós, representado a azul.

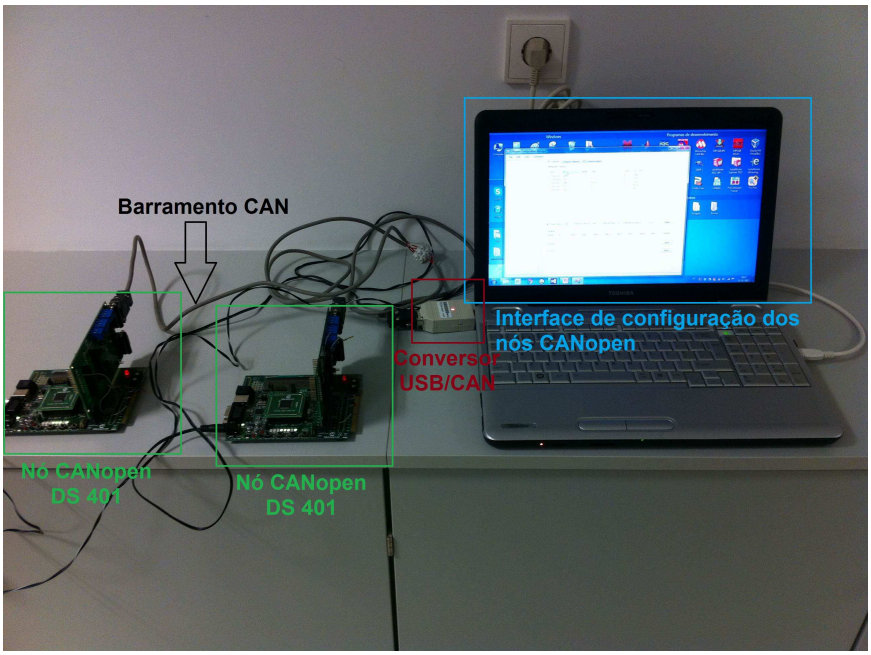

Figura 4.12: Fotografia do sistema desenvolvido.

4.3.2 Interface Gráfica

Antes de ser realizada a interface foi realizado um estudo de como poderia ser a sua arquitetura. Este estudo foi auxiliado por um programa designado por Balsamiq Mockups. Esta ferramenta permitiu projetar a arquitetura da interface antes de começar a programar, tornando possível a discutir e imaginar cenários possíveis para que quando se passasse para a fase de programação, a arquitetura da interface já tivesse definida.

A linguagem de programação para o desenvolvimento da interface para o computador foi o C# ou C Sharp, uma linguagem de programação orientada a objetos, desenvolvida pela Microsoft como parte da plataforma .NET. A sua sintaxe orientada a objetos foi baseada no C++, mas inclui influências de outras linguagens de programação, como Object Pascal ou Java. Os conhecimentos adquiridos até então passava pela programação em Visual Basic. Mas a escolha recaiu no desenvolvimento da interface em C#, uma vez que é uma linguagem com uma forte presença na automação e em comparação com o Visual Basic apresenta uma maior flexibilidade de programação e é preferível por causa das bibliotecas de classes e pelo desempenho. A afirmação de que se tratava de uma linguagem de programação com uma forte presença na automação, passa essencialmente por

algumas marcas representadas pela Bresimar, como a Beckhoff e a Beijer, apresentarem produtos nos quais podem acrescentadas novas funcionalidades através de C#.

Dicionário de objetos

A descrição pormenorizada de um dispositivo num dicionário de objetos é uma das propriedades mais relevantes na implementação deste protocolo. Como já foi referido, todos os dados importantes, parâmetros e funções de um dispositivo são fornecidas pelo dicionário de objetos através de um índice de 16 bits e um subíndice de 8 bits.

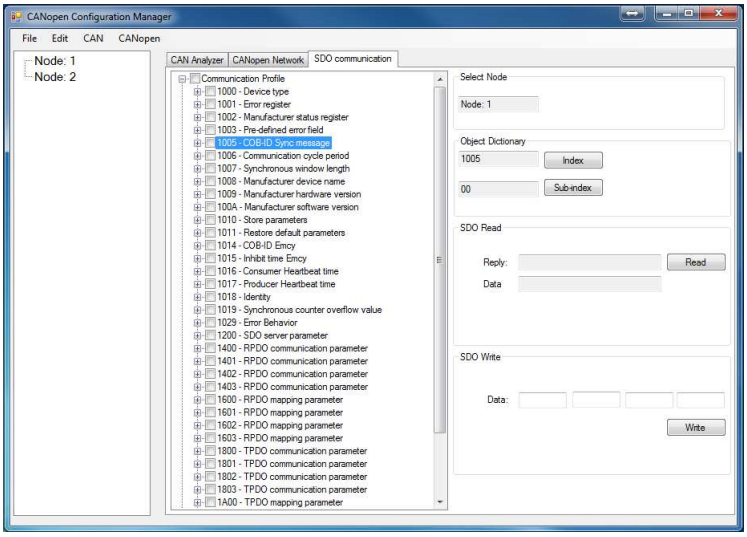

Figura 4.13: Gestão do dicionário de objetos.

Este separador da interface fornece o acesso ao dicionário de objetos dos módulos implementados. Através da seleção do nó, do índice e subíndice pudemos ler ou escrever através da comunicação SDO (ver figura 4.13). Digamos que este serviço de comunicação tem especial importância na configuração dos dispositivos. Anteriormente já foi mencionado que, esta comunicação é realizada através de duas mensagens. Uma que o servidor faz o pedido de leitura ou de escrita dos dados (ver tabela 4.7) e a outra que passa por uma resposta por parte do cliente (ver tabela 4.8), em que caso lhe tenha sido pedido uma leitura envia os dados a serem lidos e caso lhe tenha sido pedido para escrever determinados dados faculta o sucesso da escrita. Um dos requisitos mínimos da implementação do protocolo CANopen passa pela verificação do sucesso da mensagem de escrita. Através dessa mensagem o utilizador saberá se a escrita foi bem sucedida. Em caso de insucesso o nó é capaz de especificar a causa da falha (ver tabela A.4).

Basicamente este protocolo fornece dois serviço de para transferência de dados. No que toca à transferência de dados de processo em tempo-real, como anteriormente foi referido este deve ser configurado previamente. A pilha protocolar em questão permite fazer a respetiva configuração de uma maneira dinâmica, isto é, ela pode ser realizada no estado operacional. Através deste separador, pode ser por isso realizada a configuração das mensagens PDO, uma vez tendo acesso ao dicionário de objetos de um dispositivo esta comunicação pode ser parametrizada e mapeada.

Campo de arbitragem	Campo de dados							
COBID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
600h + Node ID	40h	Índice do objeto (LSB)	Índice do objeto (MSB)	Sub-índice	Campo de dados vázio			
580h + Node ID	42h, 4Fh, 4Bh, 43h	Índice do objeto (LSB)	Índice do objeto (MSB)	Sub-índice	Campo de dados			

Tabela 4.7: Tabela dos campos que compõem a trama de leitura acelerada SDO.

Campo de arbitragem	Campo de dados							
COBID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
600h + Node ID	22h, 2Fh, 2Bh, 23h	Índice do objeto (LSB)	Índice do objeto (MSB)	Sub-índice	Campo de dados			
580h + Node ID	60h, 80h	Índice do objeto (LSB)	Índice do objeto (MSB)	Sub-índice	Campo de dados			

Tabela 4.8: Tabela dos campos que compõem a trama de escrita acelerada SDO.

Gestão da rede CANopen

Neste separador da interface é possível configurar os parâmetros da rede sem que seja necessário o conhecimento sobre a composição das mensagens a enviar. Pode ser realizado a transição dos estados NMT dos dispositivos ligados ao barramento, a configuração do seu comportamento da inicialização de um nó, a atribuição da faculdade de um nó gerar um sinal síncrono, na configuração do serviço *Heartbeat* e na possibilidade de enviar comandos para que o nó possa abrir ou gravar os dados referentes à comunicação CANopen na EEPROM (ver figura 4.15).

COB ID	Byte 1	Byte 2
00h	Comando (cs)	ID do Nó (1 a 127)

Figura 4.14: Estrutura da mensagem NMT.

O protocolo CANopen fornece um serviço de gestão do estado de comunicação dos nós da rede, designado por NMT. Num sistema de controlo distribuído este serviço ganha importância no que toca à gestão do estado de comunicação da rede e do controlo dos dispositivos.

Através do envio da mensagem clarificada pela figura 4.14 podem ser mudados os estados NMT dos nodos que estejam ligados ao mesmo barramento. A mensagem NMT é composta por uma mensagem CAN com 2 bytes de dados e o identificador da mensagem é zero, identificador este que indica que é a mensagem com maior prioridade no barramento. O primeiro byte contém a especificação do comando (ver tabela 4.9 e o segundo byte contém o identificador do nó do equipamento que pretende realizar o comando. Caso o campo do identificador do nó for zero, a mensagem é dirigida a todos os nós da rede.

Desta forma, permite que a interface possa enviar diferentes comandos NMT para todos os nós ou para um nó em particular.

Função NMT	**Comando (cs)**
Start Remote Node	1
Stop Remote Node	2
Enter Pre-Operational State	80
Reset Node	81
Reset Communication	82

Tabela 4.9: Comandos para alterar os estados NMT.

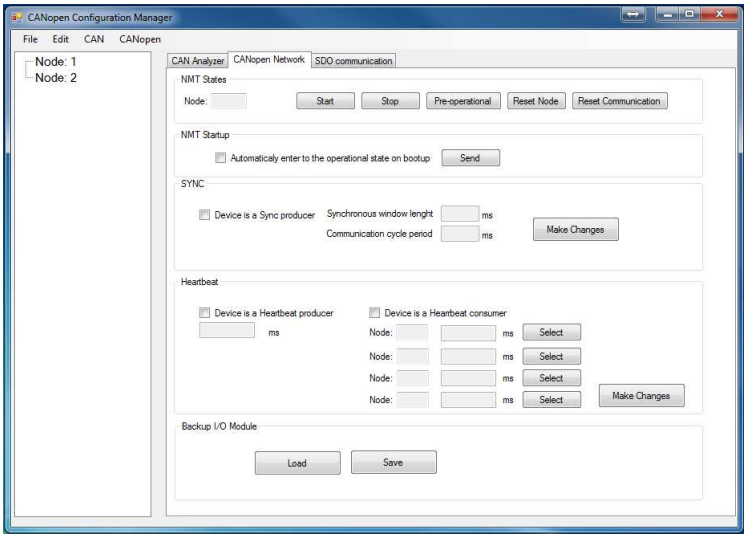

Figura 4.15: Gestão e configurações da rede CANopen.

Como foi referido anteriormente nos objetos 1F80h e 1F81h, são definidos os parâmetros que definem o comportamento ou papel na gestão da rede. Na pilha protocolar implementada nos nós, este objeto só é parcialmente implementado. Por isso através da interface podemos configurar apenas o comportamento na inicialização de cada dispositivo, isto é, se ao inicializar queremos que este transite automaticamente para o estado operacional ou não.

Índice do objecto	Sub-índice	Nome do objecto	Tipo de dados
1005H	00H	COB-ID Sync	UNSIGNED32
1006H	00H	Período do ciclo de comunicação	UNSIGNED32
1007H	00H	Comprimento da janela de sincronização	UNSIGNED32

Tabela 4.10: Entradas do dicionário de objetos para a configuração da sincronização.

Esta janela da interface permite também a configuração do serviço de sincronização, que foi anteriormente apresentado. A pilha protocolar CANopenNode implementa a faculdade de um nó gerar ou consumir um sinal síncrono. Possibilita, por exemplo, que os dois nós implementados possam trocar dados de maneira síncrona através do serviço

PDO. Por outras palavras esta janela escreve, através do serviço SDO, em três objetos responsáveis pela ativação e configuração deste serviço (ver tabela 4.10).

Na pilha protocolar dos nós é implementado o serviço *Heartbeat*, que de certa forma consiste na capacidade de produzir uma mensagem *Heartbeat* e consumir até quatro mensagens *Heartbeat*. Este serviço, como anteriormente foi explicado o seu funcionamento geral, ganha especial importância na faculdade de transmitir uma mensagem com o seu estado NMT (ver tabela 4.11). Quando é utilizada a funcionalidade de consumir uma mensagem *Heartbeat* de um nó, o nó que está a consumir essa mensagem controlará a presença deste nó, por outras palavras se o nó não enviar esta mensagem no intervalo de tempo configurado no nó consumidor, o nó consumidor deixará de estar no estado NMT operacional.

Código de estado	Estado NMT
00h	Boot-up
04h	Preparado (Stop)
05h	Operacional
7Fh	Pré-operacional

Tabela 4.11: Código de estado da mensagem *Heartbeat*.

Ao nível das mensagens enviadas para configurar este serviço, o papel da interface passa por enviar mensagens SDO que preencham os campos de dados dos objetos 1016h e/ou 1017h, descritos pela tabela 4.12.

Índice do objeto	Tipo de objeto/Sub-índice	Nome do objeto	Tipo de dados
1016h	Array	Tempo do consumidor Heartbeat	UNISIGNED32
	00h	Número de entradas	UNISIGNED32
	01-05h	Tempo do consumido Heartbeat	UNISIGNED32
1017h	Var	Tempo do produtor de Heartbeat	UNISIGNED16
	00h	Tempo do produtor de Heartbeat	UNISIGNED16

Tabela 4.12: Entradas do dicionário de objetos para a configuração do serviço *Heartbeat*.

Analisador CAN

Quando um sistema apresenta um número de dados significativo, é boa prática que se avalie as configurações que vão ser realizadas. Para avaliar a largura de banda necessária, os dados que se pretendem transmitir, o número de nós utilizados e a frequência que os dados vão ser transmitidos deverão ser projetados numa tabela. E através das equações [4.1] e [4.2], podem ser calculados alguns campos da respetiva tabela. Na primeira equação pode ser calculado os "bits por segundo", que consiste no número total de bits produzidos por segundo para transmitir uma dada mensagem e na segunda equação a largura de banda ocupada por cada variável individual [41].

$$bps = \frac{1000}{Time(ms)} \times (Bytes \times 8 + Ovr) \tag{4.1}$$

$$Percentagem_{(larguradebanda)} = \frac{bps}{Baudrate(Kbps)} \tag{4.2}$$

A coluna com os valores de Ovr, da tabela 4.13, apresenta o número de bits da mensagem PDO que não possuem dados da variável em questão. Na tabela 4.13 é apresentado um pequeno exemplo prático, a uma taxa de transmissão de 1 Mbit/s, que demonstra bem o quanto este sistema pode crescer, uma vez que no total o valor calculado para a largura de banda foi de aproximadamente 0,37%.

Nó	Variável	Bytes	Tempo(ms)	Ovr	bps	% da largura de banda
1	Entrada digital	1	100	5	130	0.13%
1	Entrada analógica	2	300	0	53,33	0.05333%
2	Entrada digital	1	100	5	130	0.13%
2	Entrada analógica	2	300	0	53,33	0.05333%

Tabela 4.13: Variáveis de uma configuração de transmissão de dados.

Um dos requisitos mínimos que se propunha para esta interface, era um separador em que auxiliasse análises deste género, bem como pudessem ser vistas todas as mensagens que estivessem a ser transmitidas através do barramento para interpretar o comportamento dos nodos ligados ao barramento e desenvolver futuras funcionalidades à interface. Por isso, neste separador é possível transmitir mensagens CAN e é possível verificar as mensagens que estão a ser transmitidas no barramento. Neste campo de leitura é apresentado o ID, o comprimento de dados, os dados transmitidos, uma contagem das mensagens com o mesmo ID e o tempo entre mensagens com o mesmo ID.

Capítulo 5

Conclusões

5.1 Conclusões

A escolha de um protocolo de comunicação como o CAN garantia fiabilidade e custos reduzidos, características já bem reconhecidas neste protocolo. Este protocolo apresentou também uma boa resposta, especialmente nos aspetos como a escalabilidade e a modularidade, principalmente por causa do protocolo de acesso ao meio, da velocidade de transmissão e do tamanho da mensagem. O protocolo CANopen permitiu definir essencialmente a camada de aplicação, que demonstrou sobretudo vantagens no que toca à definição da priorização das mensagens, nos serviços de comunicação e de gestão da rede fornecidos e pela forma estruturada com que apresenta os dados e funções do protocolo de comunicação.

Cada vez mais a tendência passa para que as comunicações no meio industrial sejam cobertas pelas redes Ethernet, porque além deste tipo de redes já serem dominantes nos níveis superiores, cada vez mais apresentam diferentes soluções para o nível de campo. Como os fluxos de informação não circulam apenas entre dispositivos do mesmo nível, com a respetiva proliferação das redes Ethernet, torna-se uma vantagem o CANopen disponibilizar de uma forma estruturada a informação de processos, ou em sentido contrário, execução de ordens, provenientes de protocolos com melhores características para níveis superiores. Por outras palavras, apesar do protocolo CANopen possa não ser uma preferência a níveis mais elevados na hierarquia das comunicações, este apresenta características importantes para que se possa fazer interação com outros protocolos de comunicação. O dicionário de objetos é a principal vantagem apresentada ao disponibilizar a informação de um dispositivo de uma forma organizada e de fácil acesso. Por isso, é certo que com este protocolo podem ser desenvolvidas soluções modulares em que se queiram apresentar uma característica, muito vantajosa para sistemas incorporados distribuídos, tal como a escalabilidade, a fiabilidade e o controlo.

O protocolo CANopen é um protocolo cada vez com mais provas dadas para alguns requisitos da indústria, apesar de não se encontrarem muitos dispositivos e soluções com este protocolo de comunicação, que é aberto e apresenta baixos custos. Por isso, nesta dissertação foi apresentado um trabalho de pesquisa e implementação do protocolo CANopen através da pilha protocolar CANopenNode.

Quanto à pilha protocolar, a escolha recaiu sobre o CANopenNode, porque foi a que apresentou mais funcionalidades e melhores características para o desenvolvimento futuro com outras funcionalidades. Com esta pilha protocolar é possível apresentar soluções de

automação de baixo custo, e que pode ser implementada em vários microcontroladores.

Este trabalho tornou-se também vantajoso, pois possibilitou o estudo de um protocolo com um serviço de transferência de dados em tempo-real bastante interessante e que até já foi transposto em outros protocolos de comunicação que viriam a ser introduzidos depois deste. No que respeita à transferência de dados, basicamente existem duas diferentes maneiras de a executar. O serviço SDO, que tem por base uma comunicação cliente/servidor e permite o endereçamento direto de um objeto usando o índice de 16 bits e sub-índice de 8 bits. Este serviço de comunicação assíncrono é o mais indicado para a configuração de um dispositivo, porque a transferência de dados por este serviço tem baixa prioridade e permite transferência de maior carga de dados. O serviço PDO fornece uma transmissão eficiente e de tempo-real de dados, muito por causa do modelo de transferência de dados ser do tipo produtor/consumidor. Uma mensagem PDO pode conter valores de mais que uma entrada do dicionário de objeto, mas o conteúdo de uma PDO tem que ser definido, antes da transmissão ser ativada, bem como o comprimento de dados pode ser menor que 8 bytes mas terá de ser múltiplo de 1 byte. O protocolo CANopen permite que um dispositivo possa especificar até 512 PDO para receber e transmitir, dentro das limitações do sistema, como a memória, poder de processamento, ou da rede, devido ao número de identificadores disponíveis CAN. Mas cada nó implementado neste trabalho pode ter no máximo quatro PDOs de transmissão e quatro de recepção disponíveis para transmissão e receção de dados, e também não se sentiu necessidade de aumentar este número. Este modelo permite que a troca de dados, uma vez configurada, necessite apenas de a transmissão de uma mensagem. Portanto este tipo de comunicação diminuí a sobrecarga no barramento, mas leva a que não seja facultada o sucesso ou insucesso da transferência de dados.

No que toca à interface gráfica que foi desenvolvida ficou ainda por serem explorados mais funcionalidades para tirar o máximo partido do protocolo de comunicação implementado nos nodos. Contudo no desenvolvimento desta foi tido em conta a possibilidade de crescimento e aperfeiçoamento. A ferramenta Balsamiq Mockups foi essencial para a estruturação e apresentação da funcionalidades previstas. Do que foi apresentado através desta ferramenta ficou por ser explorado a configuração do serviço PDO. A idealização desta configuração foi umas das tarefas em que foi apresentada maior sensibilidade no seu tratamento, uma vez que se pretendia tirar partido da maior flexibilidade de configuração dada por este serviço de transmissão de dados. Por outras palavras, este serviço de transmissão de dados permite serem realizadas um leque de possibilidades de configuração. Contudo na idealização da interface houve uma ponderação cuidada para tornar a configuração amigável para o utilizador sem limitar a respetiva flexibilidade.

5.2 Trabalho futuro

Do trabalho desenvolvido e do ponto de vista das necessidades mais importantes, deixo para trabalho futuro que a configuração do serviço PDO deva ser tratada em redor de um nó e diferenciando na interface se estamos a tratar da configuração da transmissão ou da recepção desse mesmo nó. Toda esta configuração poderia recorrer ao armazenamento dos dados da configuração num formato *eXtensible Markup Language* (XML), formato que permite boa manipulação crescente e flexível, para que seja possível apresentar as configurações que foram realizadas através da interface.

Deixo também como trabalho futuro uma melhor abordagem ao dicionário de objetos. A interface apresenta um dicionário de objetos estático e não grava esses dados num ficheiro. Nos *softwares* que foram acessíveis para serem explorados verificou-se que os dados do dicionário de objetos eram armazenados em ficheiros EDS. Apesar disso deixo como sugestão para trabalho futuro, de modo a facilitar a sua implementação em C#, o armazenamento deste dados num ficheiro XML. Esta funcionalidade facilitaria a interação entre a interface e os dispositivos ligados ao barramento, bem como à introdução de dispositivos com funcionalidades diferentes.

Bibliografia

[1] S. Pinheiro. Impacto da Rede de Comunicação em Sistemas de Controlo Distribuídos. Master's thesis, Universidade de Aveiro, 2005.

[2] Tiago Costa Gonçalves. Implementação de uma Rede de Domótica Baseada em Ethernet e CANopen. Master's thesis, Universidade de Aveiro, 2010.

[3] Eduardo Manuel de Médicis Tovar. Redes de Comunicação Industriais do Tipo Field Bus, Integração em Ambiente CIM. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 1994.

[4] Rui Jorge Cunha Sancho. Sistemas de controlo distribuídos baseado em barramento CAN. Master's thesis, Universidade de Aveiro, 2009.

[5] P. Leitão; F. Restivo. A Layered Approach to Distributed Manufacturing. 1999.

[6] M. Groover. *Automation, Production Systems and CIM.* Prentice-Hall, 1987.

[7] P. Leitão. *An agile and adaptive holonic architecture for manufacturing control.* PhD thesis, Faculdade de Engenharia da Universidade do Porto, 2004.

[8] J. Black. *The Design of the Factory with a Future.* McGraw Hill, 1991.

[9] J. Womack; D. T. Jones; D. Roos. *The Machine that Changed the World.* MIT Press, Cambridge, 1990.

[10] P. T. Kidd. *Agile Manufacturing: Forging New Frontiers.* Addison-Wesley, 1994.

[11] S. Goldman; R. Nagel; K. Preiss. *Agile Competitors and Virtual Organisations.* Van Nostrand Reinhold, New York, 1995.

[12] B. Pine; S. Davis. *Mass customization: The new frontier in business competition.* Harvard Business Review Press, 1999.

[13] U. Rembold; B. Nnaji; A. Storr. *Addison-Wesley.* Computer Integrated Manufacturing and Engineering, 1993.

[14] Nadia Gomes. Controle e planejamento da produção através do sistema de informação atual. Master's thesis, Universidade Candido Mendes, Rio de Janeiro, Brasil, 2010.

[15] A. W. Scheer. *CIM: Towards the Factory of the Future.* Springer-Verlag, 1994.

[16] V. H. Russomano. *Planejamento e Controle da Produção*. Ed. Pioneira, São Paulo, Brasil, 1995.

[17] S. B. Zaccarelli. *Programação e Controle da Produção*. Ed. Pioneira, São Paulo, Brasil, 1979.

[18] Daifuku AGV image. [Online] Disponível em `http://www.daifukumexico.com/products/73/338/894/Manufacturing-Distribution/Conveyor-Vehicle-Systems/Automatic-Guided-Vehicle-AGV`. Acedido a 6/3/2013.

[19] UNARCORACK Pallet Rack and Engineered Systems. [Online] Disponível em: `http://www.unarcorack.com/portfoliocategories/asrs/`. Acedido a 6/3/2013.

[20] William W. Luggen. *Flexible Manufacturing Cells and Systems*. Prentice Hall.

[21] H. Schumny. Fieldbuses in measurement and control. *Computer Standards and Interfaces*, 19:295 – 304, 1998.

[22] R. Patzke. Fieldbus basics. *Computer Standards and Interfaces*, 19:275 – 293, 1998.

[23] L. Almeida. *Flexibility and Timeliness in Fieldbus-Based Real-Time Systems*. PhD thesis, Universidade de Aveiro, 1999.

[24] A. L. L. Antunes. *Algoritmos de Controlo Distribuído em Sistemas Baseados em Microprocessadores*. PhD thesis, Universidade de Aveiro, 2008.

[25] W. Lawrenz. *CAN System Engineering: From Theory to Practical Applications*. Springer-Verlag, 1997.

[26] S. Yang J. Tsai, Y. Bi and R. Smith. *Distributed Real-Time Systems: Monitoring, Visualization, Debbuging and Analysis*. Wiley-Interscience, 1996.

[27] H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academis Publishers, 1997.

[28] A. Ray. Network access protocols for real-time distributed control systems. *IEEE Transactions on industry applications*, 24:897 – 904, 1988.

[29] J. W. S. Liu. *Real-Time Systems*. Prentice Hall, 2000.

[30] P. Pedreiras. *Supporting Flexible Real-Time Communication on Distributed Systems*. PhD thesis, Universidade de Aveiro, 2003.

[31] J.P. Thomesse. A review of the fieldbuses. *Annual Reviews in Control*, 22:35 – 45, 1998.

[32] Altera Industrial Fieldbus Protocols. [Online] Disponível em: `http://www.altera.com/end-markets/industrial/automation/ethernet/protocols/ind-fieldbuses.html`. Acedido a 15/4/2013.

[33] A. B. Lugli; M. M. D. Santos. *Sistemas Fieldbus para Automação Industrial*. Ed. Érica, 2009.

[34] S. Mackay; E. Wright; D. Reynders; J. Park. *Practical Industrial Data Networks: Design, Installation and Troubleshooting*. IDC Technology. Newnes, 1 edition, 2004.

[35] M. Figueiredo. *Controlo Distribuído de Plataformas para Experiências de Mecatrónica*. PhD thesis, Universidade de Aveiro, 2008.

[36] Modbus Technical Resources. [Online] Disponível em: `http://www.modbus.org/tech.php`. Acedido a 8/4/2013.

[37] Anybus EtherCAT. [Online] Disponível em: `http://www.anybus.com/technologies/ethercat_tech.shtml`. Acedido a 8/4/2013.

[38] L. Seno; S. Vitturi; C. Zunino. Real Time Ethernet Networks Evaluation Using Performance Indicators. 2009.

[39] Anybus Ethenet Powerlink. [Online] Disponível em: `http://www.anybus.com/technologies/powerlink_tech.shtml`. Acedido a 8/4/2013.

[40] CiA specifications. [Online] Disponível em `http://www.can-cia.org/index.php?id=440`. Acedido a 24/6/2013.

[41] O. Pfeiffer; A. Ayre; C. Keydel. *Embedded Networking with CAN and CANopen*. Copperhill Media Corporation, 1 edition, 2008.

[42] CANopen Solutions. [Online] Disponível em `http://www.canopensolutions.com/index.html`. Acedido a 25/6/2013.

[43] Konrad Etshberger. *Controller Area Network*. IXXAT Automation GmbH, 2001.

[44] YAMAR. *DCAN250 - CAN over Battery Power Line Communication*, 2007.

[45] Understanding Microchip CAN Module Bit Timing. [Online] Disponível em: `http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en011947`. Acedido a 2/5/2013.

[46] Minkoo Kang; Kiejin Park; Bongjun Kim1. PDO Packing Mechanism for Minimizing CANopen Network Utilization. *Industrial Electronics*, pages 1516 – 1519, 2008.

[47] K. Tindell; A. Burns; A. J. Wellings. Calculating Controller Area Network (CAN) Message Response Times. *Control Engineering Practice*, 3:1163 – 1169, 1995.

[48] K. Tindell; A. Burns. Guaranteed Message Latencies for Distributed Safety-critical Hard Real-time Networks. Technical report, Department of Computer Science, University of York, 1994.

[49] CAN/LIN PICtail (Plus) Daughter Board. [Online] Disponível em `http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en535527`. Acedido a 2/5/2013.

[50] High-Speed CAN Transceiver MCP2551. [Online] Disponível em `http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010405`. Acedido a 3/5/2013.

[51] CiA Draft Recommendation Proposal 303-1. *CANopen Cabling and Connector Pin Assignment*. CiA.

[52] PIC32MX5XX/6XX/7XX. [Online] Disponível em `http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en545660`. Acedido a 12/6/2013.

[53] Peak PCAN-USB. [Online] Disponível em `http://www.peak-system.com/PCAN-USB.199.0.html?&L=1`. Acedido a 12/7/2013.

[54] ESD CAN/USB2.0 Interface. [Online] Disponível em `http://www.carambola.cc/`. Acedido a 12/7/2013.

[55] MHS TINY-CAN. [Online] Disponível em `http://www.mhs-elektronik.de/index.php?module=content&action=show&page=tinycan_uebersicht`. Acedido a 15/7/2013.

[56] ESD CAN/USB2.0 Interface. [Online] Disponível em `http://esd.eu/en/products/can-usb2`. Acedido a 15/7/2013.

[57] IXXAT-USB/CAN Interface. [Online] Disponível em `http://www.ixxat.com/usb-to-can-compact-interface_en.html`. Acedido a 15/7/2013.

[58] Zanthic Tecnhologies. [Online] Disponível em `http://www.zanthic.com/products.htm`. Acedido a 16/7/2013.

[59] Lawicel CAN/USB. [Online] Disponível em `http://www.canusb.com/`. Acedido a 15/7/2013.

[60] KVASER CAN/USB. [Online] Disponível em `http://www.kvaser.com/en/products/can/usb.html`. Acedido a 16/7/2013.

[61] EMS hardware CAN/USB. [Online] Disponível em `http://www.ems-wuensche.com/product/datasheet/html/can-usb-adapter-converter-interface-cpcusb.html`. Acedido a 16/7/2013.

Apêndice A

Dados padrão do protocolo CANopen

A.1 Camada física

Taxa de transmissão	Máximo comprimento do barramento	Bit Time	Time Quanta por bit	Comprimento de um TQ	Ponto de amostragem
1 Mbp/s	25 m	1 μs	8	125 ns	6 TQ (75%)
800 Kbp/s	50 m	1.25 μs	10	125 ns	8 TQ (80%)
500 Kbp/s	100 m	2 μs	16	125 ns	14 TQ (87.5%)
250 Kbp/s	250 m	4 μs	16	250 ns	14 TQ (87.5%)
125 Kbp/s	500 m	8 μs	16	500 ns	14 TQ (87.5%)
50 Kbp/s	1 km	20 μs	16	1.25 μs	14 TQ (87.5%)
20 Kbp/s	2.5 km	50 μs	16	3.125 μs	14 TQ (87.5%)
10 Kbp/s	5 km	100 μs	16	6.25 μs	14 TQ (87.5%)

Tabela A.1: Recomendações do *Bit Timing* CANopen.

A.2 *Comunication Objects*

Objetos de comunicação	**COB-ID**	**Papel do nó escravo**
Controlador NMT	0	Apenas recebe
Sync	80	Apenas recebe
Emergency	080 + ID	Transmite
PDO	180 + ID	Transmite PDO (1)
	200 + ID	Recebe PDO (1)
	280 + ID	Transmite PDO(2)
	300 + ID	Recebe PDO (2)
	380 + ID	Transmite PDO(3)
	400 + ID	Recebe PDO (3)
	480 + ID	Transmite PDO (4)
	500 + ID	Recebe PDO (4)
SDO	580 + ID	Transmite
	600 + ID	Recebe

Tabela A.2: Objetos de comunicação.

A.3 *Standardized device application profiles*

Número do perfil	**Classe dos equipamentos**
CiA 401	Generic I/O Modules
CiA 402	Drives and Motion Control
CiA 404	Measuring devices and Closed Loop Controllers
CiA 405	IEC 61131-3 Programmable Devices
CiA 406	Rotating and Linear Encoders
CiA 408	Hydraulic Drives and Proportional Valves
CiA 410	Inclinometers
CiA 412	Medical Devices
CiA 413	Truck Gateways
CiA 414	Yarn Feeding Units (Weaving Machines)
CiA 415	Road Construction Machinery
CiA 416	Building Door Control
CiA 417	Lift Control Systems
CiA 418	Battery Modules
CiA 419	Battery Chargers
CiA 420	Extruder Downstream Devices
CiA 422	Municipal Vehicles - CleANopen
CiA 423	Railway Diesel Control Systems
CiA 424	Rail Vehicle Door Control Systems
CiA 425	Medical Diagnostic Add-on Modules
CiA 445	RFID Devices

Tabela A.3: Tabela como os perfis de equipamento CANopen.

A.4 Comunicação SDO

A.4.1 Código de falha

Código de falha	Descrição
0503 0000	*Toogle* bit não alternou
0504 0000	Tempo limite no serviço SDO excedido
0504 0001	Comando não é válido
0504 0002	Comprimento do bloco inválido
0504 0003	Número de sequência inválido
0503 0004	Erro na validação do campo CRC
0503 0005	Sem memória
0601 0000	Sem suporte de acesso ao objeto
0601 0001	Tentativa de leitura num objeto só de escrita
0601 0002	Tentativa de escrita num objeto só de leitura
0602 0000	O objeto não existe no dicionário de objetos
0604 0041	O objeto não pode ser mapeado no PDO
0604 0042	O número ou o comprimento dos objetos para serem mapeado excedem o comprimento PDO
0604 0043	Incompatibilidade do parâmetro
0604 0047	Incompatibilidade interna no dispositivo
0606 0000	Falha no acesso devido ao *hardware*
0607 0010	Não corresponde ao comprimento do parâmetro de serviço
0607 0012	Excedeu o comprimento especificado do parâmetro de serviço
0607 0013	Comprimento reduzido do parâmetro de serviço
0609 0011	Sub-índice não existe
0609 0030	Valor inválido para o parâmetro
0609 0031	Valor do parâmetro escrito muito baixo
0609 0032	Valor do parâmetro escrito muito alto
0609 0036	Valor máximo encontra-s a baixo do valor minimo
0800 0000	Erro geral
0800 0020	Os dados não podem ser transferidos ou armazenados na aplicação
0800 0021	Os dados não podem ser transferidos ou armazenado na aplicação por causa do controlo local
0800 0022	Os dados não podem ser transferidos ou armazenados na aplicação por causa do estado do dispositivo
0800 0023	Falha na criação dinâmica do objeto de dicionário ou não há objeto de dicionário
0800 0024	Dados não são válidos

Tabela A.4: Códigos de falha do serviço de transferência de dados SDO

A.4.2 Leitura acelerada

COB ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
(600h + Node ID)	Comando	Índice do dicionário de objetos		Subíndice	Reservado (zero)			

Figura A.1: Formato da mensagem de pedido de leitura de forma acelerada

Comando (aaab bbbb)	0x40
aaa	010 (ccs = 2: inicialização do *upload*)
bbbbb	00000 (não é usado)

Tabela A.5: Byte de comando para pedido de leitura.

COB ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
(580h + Node ID)	Resposta	Índice do dicionário de objetos		Subíndice	Dados			

Figura A.2: Formato da mensagem de resposta ao pedido de leitura de forma acelerada

Resposta (aaabcces)	0x4F - 0x43
aaa	010 (scs = 2; inicialização do *upload*
b	0 (não é usado)
cc	(número de bytes onde o campo de dados contém os dados)

Tabela A.6: Byte de comando para resposta ao pedido de leitura acelerada.

A.4.3 Escrita acelerada

COB ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
(600h + Node ID)	Comando	Índice do dicionário de objetos		Subíndice	Dados			

Figura A.3: Comunicação SDO: Formato das mensagem de escrita acelerada

Comando (aaabcces)	0x2F - 0x23
aaa	001 (scs = 1; inicialização do *download*)
b	0 (não é usado)
cc	(número de bytes onde o campo de dados contém os dados)
e	1 (transferência acelerada)
s	1 (tamanho dos dados é indicado)

Tabela A.7: Byte de comando da mensagem de escrita acelerada.

COB ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
(580h + Node ID)	Resposta	Índice do dicionário de objetos		Subíndice	Reservado (zeros)			

Figura A.4: Comunicação SDO: Formato da mensagem de confirmação de escrita

Comando (aaab bbbb)	0x60
aaa	011 (ccs = 3: inicialização do *download*)
bbbbb	00000 (não é usado)

Tabela A.8: Byte de comando para confirmar a escrita.

A.5 Comunicação PDO

A.5.1 Configuração da comunicação PDO

Sub-Índice	Conteúdo	Tipo de dados	Tipo de acesso
0	Valor máximo de sub-índice	UNSIGNED8	ro
1	COB-ID	UNSIGNED32	rw
2	Tipo de transmissão	UNSIGNED8	rw
3	Tempo de inibição (ms)	UNSIGNED16	rw
4	Não é usado	UNISIGNED8	rw
5	Temporizador para a transmissão por evento	UNSIGNED16	rw
6	Valor Sync	UNSIGNED8	rw

Tabela A.9: Parâmetros de transmissão PDO.

Sub-Índice	Conteúdo	Tipo de dados	Tipo de acesso
0	Valor máximo de sub-índice	UNSIGNED8	ro
1	COB-ID	UNSIGNED32	rw
2	Tipo de transmissão	UNSIGNED8	rw

Tabela A.10: Parâmetros de recepção PDO.

Valor (dec)	Tipo de transmissão
0	Síncrona
1-240	Síncrona (transmite depois depois de receber "n"mensagens Sync)
241-251	não é usado
252-253	Remota
254	Específico do fabricante
255	Assíncrono

Tabela A.11: Tipos de transmissão PDO.

A.6 Descrição do serviço emergência

A.6.1 Códigos de erro

Código de erro	Descrição do erro
00xx	Erro de reset/Sem erro
10xx	Erro genérico
20xx	Corrente
21xx	Corrente (entradas do dispositivo)
22xx	Corrente (interior do dispositivo)
23xx	Corrente (saídas do dispositivo)
30xx	Tensão
31xx	Tensão da rede eléctrica
32xx	Tensão no interior do dispositivo
33xx	Tensão de saída
4xxx	Temperatura
41xx	Temperatura ambiente
42xx	Temperatura do dispositivo
50xx	Hardware
60xx	Software
61xx	Internal software
62xx	User software
63xx	Data set
70xx	Módulos adicionais
81xx	Monitorização
8110	Comunicações
8120	CAN overrun
8130	Erro passivo
8140	Heartbeat ou Life Guard
8140	Recuperação do congestionamento do barramento
8150	Colisão CAN-ID
82xx	Erro no protocolo de comunicação
8210	CAN overrun (perdeu objectos)
8220	CAN no modo passivo de erro
8230	Erro no serviço Heartbeat
8240	Comprimento de dados SYNC inesperado
8250	Tempo limite RPDO
90xx	Erro externo
F0xx	Funções adicionais
FFxx	Específico do dispositivo

Tabela A.12: Descrição dos códigos de erro.

A.6.2 Codificação da causa de erro

Bit	Causa do erro
0	Erro genérico
1	Corrente
2	Voltagem
3	Temperatura
4	Erro de comunicação
5	Específico do dispositivo
6	Reservado
7	Específico do fabricante

Tabela A.13: Tabela de codificação do campo que descreve a causa de erro.

Apêndice B

Código de aplicação do PIC32 na Explorer16

```
/*****************************************************************
 *              General Code
 *****************************************************************
 * FileName:        General.c
 * Dependencies:
 * Processor:       PIC32MX795F512L
 * Complier:        Microchip XC32
 * Company:         Microchip Technology, Inc.
 *
 *
 * Author               Date        Comment
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * CANopenNode
 * João Coelho          20/05/2013  Version Learning
 *****************************************************************/

#include "CANopen.h"
#include "general.h"

//Global variables
   extern CO_t *CO;                          //pointer to external CANopen object


/*******************************************************************************

#if ODL_errorStatusBits_stringLength < 10
      #error Error in Object Dictionary!
#endif

#define ERROR_TEST1_INFORMATIVE                     0x30, 0x1000
#define ERROR_TEST2_CRITICAL                        0x40, 0x5000


void ConfigurationsBits_ClockSettings(void)
{
    //Configuration bits
   #pragma config FVBUSONIO = OFF       //USB VBUS_ON Selection (OFF = pin is contr
       olled by the port function)
   #pragma config FUSBIDIO = OFF        //USB USBID Selection (OFF = pin is control
       led by the port function)
   #pragma config UPLLEN = OFF          //USB PLL Enable
   #pragma config UPLLIDIV = DIV_12     //USB PLL Input Divider
   #pragma config FCANIO = ON           //CAN IO Pin Selection (ON = default CAN IO
        Pins)
```

```
    #pragma config FETHIO = OFF          //Ethernet IO Pin Selection (ON = default E
        thernet IO Pins)
    #pragma config FMIIEN = OFF          //Ethernet MII Enable (ON = MII enabled)
    #pragma config FSRSSEL = PRIORITY_7 //SRS (Shadow registers set) Select
    #pragma config POSCMOD = XT          //Primary Oscillator //HS
    #pragma config FSOSCEN = OFF         //Secondary oscillator Enable
    #pragma config FNOSC = PRIPLL        //Oscillator Selection //
    #pragma config FPLLIDIV = DIV_2      //PLL Input Divider //
    #pragma config FPLLMUL = MUL_16      //PLL Multiplier  //MUL_20
    #pragma config FPLLODIV = DIV_1      //PLL Output Divider Value //
    #pragma config FPBDIV = DIV_2        //Bootup PBCLK divider //DIV_1
    #pragma config FCKSM = CSDCMD        //Clock Switching and Monitor Selection
    #pragma config OSCIOFNC = OFF        //CLKO Enable
    #pragma config IESO = OFF            //Internal External Switch Over
    #pragma config FWDTEN = OFF          //Watchdog Timer Enable //
    #pragma config WDTPS = PS1024        //Watchdog Timer Postscale Select (in milli
        seconds)
    #pragma config CP = OFF              //Code Protect Enable
    #pragma config BWP = ON              //Boot Flash Write Protect
    #pragma config PWP = PWP256K         //Program Flash Write Protect
//    #pragma config ICESEL = ICS_PGx1     //ICE/ICD Comm Channel Select
    #pragma config DEBUG = ON            //Background Debugger Enable

}


/*****************************************************************************/
void programStart(void){

    /*Initialize pins for Switch*/
    PORTSetPinsDigitalIn(IOPORT_B, BIT_0 | BIT_1 | BIT_3);
    PORTSetPinsDigitalIn(IOPORT_F, BIT_4 | BIT_5);
    PORTSetPinsDigitalIn(IOPORT_D, BIT_14 | BIT_15);
    mPORTBClearBits(BIT_0 | BIT_1 | BIT_3 );
    mPORTDClearBits(BIT_14 | BIT_15);
    mPORTFClearBits(BIT_4 | BIT_5);
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    //Enable change notice, enable discrete pins and weak pullups
    mCNOpen(CONFIGCN, PINSCN, PULLUPSCN);

    //Initialize two CAN led diodes
    #define CAN_RUN_LED         LATAbits.LATA0
    #define CAN_ERROR_LED       LATAbits.LATA1
    TRISACLR = 0xFF;
    CAN_RUN_LED = 0;        CAN_ERROR_LED = 1;

    /* Botões */
    //Initialize digital input - D
    TRISDbits.TRISD6 = 1;
    TRISDbits.TRISD7 = 1;
    TRISDbits.TRISD13 = 1;

    //Initialize Timer 3 for PWM
    /* Open Timer3 with Period register value */
    OpenTimer3(T3_ON, 0x550);
    OpenOC1( OC_ON | OC_TIMER_MODE16 | OC_TIMER3_SRC | OC_PWM_FAULT_PIN_DISABLE |
        OC_SINGLE_PULSE  , 0x550, 0x550);

    //Initialize ADC 10 - AN2
    // Configure and enable the ADC
    CloseADC10();     // ensure the ADC is off before setting the configuration
    // AD1CHS: ADC Input Select Register - configure to sample AN2
    SetChanADC10(ADC_CH0_NEG_SAMPLEA_NVREF | ADC_CH0_POS_SAMPLEA_AN2 | ADC_CH0_NEG
        _SAMPLEB_NVREF | ADC_CH0_POS_SAMPLEB_AN4);
    OpenADC10( ADCON1, ADCON2, ADCON3, ADCSSL, ADPCFG ); // configure ADC using pa
        rameter define above
```

```
    EnableADC10(); // Enable the ADC
    while ( ! mAD1GetIntFlag() );

}


/*****************************************************************************/
void communicationReset(void){
   CAN_RUN_LED = 0; CAN_ERROR_LED = 0;
   OD_writeOutput8Bit[0] = 0;
   OD_writeOutput8Bit[1] = 0;
   OD_writeAnalogueOutput16Bit[0] = 0;
}


/*****************************************************************************/
void programEnd(void){
    /* Close Output Compare */
    CloseOC1();

    /* LED */
    CAN_RUN_LED = 0; CAN_ERROR_LED = 0;
}


/*****************************************************************************/
void programAsync(unsigned int timer1msDiff){

   //Leds for verify some states of this node
   CAN_RUN_LED = LED_GREEN_RUN(CO->NMT); //Operational CAN mode
   CAN_ERROR_LED = LED_RED_ERROR(CO->NMT); //Error CAN mode

   //Is any application critical error set?
   //If error register is set, device will leave operational state.
   if(CO->EM->errorStatusBits[8] || CO->EM->errorStatusBits[9])
      *CO->EMpr->errorRegister |= 0x20;
}


/*****************************************************************************/
void program1ms(void)
{
    UNSIGNED8 leds, buttons;
    UNSIGNED16  PWM;

    //Leds for write output 8 bit on Explorer16.
    leds = OD_writeOutput8Bit[0];
    LATAbits.LATA5 = (leds&0x20) ? 1 : 0; //USE Button
    LATAbits.LATA6 = (leds&0x40) ? 1 : 0; //USE Button
    LATAbits.LATA7 = (leds&0x80) ? 1 : 0; //USE Button


    //Leds for verify some states of this node
    //Verify operating state of this node
    LATAbits.LATA2 = (CO->NMT->operatingState == CO_NMT_OPERATIONAL) ? 1 : 0;
    //Verify operating state of monitored nodes
    LATAbits.LATA3 = (CO->HBcons->allMonitoredOperational) ? 1 : 0;
    //Verify operating state of produtor HB
    LATAbits.LATA4 = (CO->NMT->HBproducerTimer) ? 1 : 0;


    //Buttons for read input 8 bit on Explorer16
    buttons = 0;
    if(!PORTDbits.RD6)  buttons |= 0x80; //S3 -> LED7
    if(!PORTDbits.RD7)  buttons |= 0x40; //S6 -> LED6
    if(!PORTDbits.RD13) buttons |= 0x20; //S4 -> LED5
    OD_readInput8Bit[0] = buttons;
```

```
    //Potenciometer for read analogue input 16 bit
    OD_readAnalogueInput16Bit[0] = ReadADC10(0);

    //Read temperature 2108h
    OD_temperature[0] = ReadADC10(1);

    //PWM in pin RD0
    PWM = (OD_writeAnalogueOutput16Bit[0] * 0x550) / 0x03FF;
    SetDCOC1PWM(PWM);

#ifdef USE_LCD
    if(Count == 0)
    {
        Count++;
    }
    else if(Count > 5000)
    {
        Count = 0;
    }
    else
        Count++;
#endif
}


/******************************************************************************/
//Function passes data to SDO server on testDomain variable access
UNSIGNED32 ODF_testDomain( void       *object,
                           UNSIGNED16  index,
                           UNSIGNED8   subIndex,
                           UNSIGNED16 *pLength,
                           UNSIGNED16  attribute,
                           UNSIGNED8   dir,
                           void       *dataBuff,
                           const void *pData){
   //pData is not valid here

    if(dir == 0)
    {
        //reading object dictionary
        UNSIGNED16 i;
        UNSIGNED16 dataSize = 889; //889 = 127*7 = 0x379 = maximum block transfer

        //verify max buffer size
        if(dataSize > CO_OD_MAX_OBJECT_SIZE) //default is 32. Redefine macro in CO
            _driver.h to 889.
        return 0x06040047L;  //general internal incompatibility in the device

        //fill buffer
        for(i=0; i<dataSize; i++)
             ((UNSIGNED8*) dataBuff)[i] = i+1;

        //indicate length
        *pLength = dataSize;

        //return OK
        return 0;
    }

    else
    {
        //writing object dictionary
        UNSIGNED16 i;
        UNSIGNED16 err = 0;
        UNSIGNED16 dataSize = *pLength;
```

```
        //do something with data, here just verify if they are the same as above
        //printf("\nData received, %d bytes:\n", dataSize);
        for(i=0; i<dataSize; i++)
        {
            UNSIGNED8 b = ((UNSIGNED8*) dataBuff)[i];
            if(b != ((i+1)&0xFF)) err++;
            //printf("%02X ", b);
        }

        if(err) return 0x06090030; //Invalid value for parameter (download only).

        //return OK
        return 0;
    }
}

#ifdef USE_LCD
/*
 * Platform: Explorer-16 with PIC32MX PIM
 *
 * Features demonstrated:
 *    - Timer configuration
 *    - PMP functions
 *    - LCD control
 *
 * Description:
 *      This program writes text to the Explorer-16 LCD display using
 *      the PMP library.
 *
 * Notes:
 *    - The Explorer-16 LCD is compatible with the standard HD44780
 *      instruction set. The specific delays and instructions needed
 *      for proper LCD functionality can be found online.
    // Delay needed for LCD to process data
 */

void Delay_ms(unsigned long x)
{
        WriteTimer1(0); // Clear the timer
        while (TMR1 < (315 * x)); // 3.2us x 315 = (~1ms delay * desired value)
}

// LCD initialization sequence
void initializeLCD(void)
{
    mPMPOpen(CONTROL, MODE, PORT, INTERRUPT); // Initialize PMP using above settin
        gs

    OpenTimer1(CONFIGLCD, PERIODLCD); // Enable Timer 1
    Delay_ms(30); // LCD needs 30ms to power on, perform startup functions, etc

    PMPSetAddress(CMDREG); // Access the LCD command register
    PMPMasterWrite(0x38); // LCD Function Set - 8-bit interface, 2 lines, 5*7 Pixe
        ls
    Delay_ms(1); // Needs a delay to complete

    PMPMasterWrite(0x0C); // Turn on display (with cursor hidden)
    Delay_ms(1); // Needs a delay to complete

    PMPMasterWrite(0x01); // Clear the display
    Delay_ms(2); // Needs a delay to complete

    PMPMasterWrite(0x06); // Set text entry mode - auto increment, no shift
    Delay_ms(1); // Needs a delay to complete
}
```

```
// Write a byte of data to either of the two LCD registers (DATAREG, CMDREG)
void writeToLCD(int reg, char c)
{
    Delay_ms(1); // needed between each character write
    PMPSetAddress(reg); // Select either 'DATAREG' or 'CMDREG'
    PMPMasterWrite(c); // Write the byte to selected register
}

// Used to write strings to the LCD
void writeString(char *string)
{
    while (*string) { // Keep going until string ends
        writeToLCD(DATAREG, *string++); //Send characters one by one
    }
}

// Set the LCD cursor position to line two
void newLine(void)
{
    writeToLCD(CMDREG, 0xC0); // Cursor address 0x80 + 0x40 = 0xC0
}

void LineOne(void)
{
    writeToLCD(DATAREG, 0xC0);
}


void writeNumber(UNSIGNED16 number)
{
    char buffer[sizeof(number)];
    sprintf(buffer,"%d", number);
    writeString(buffer);
}

#endif
```

Apêndice C

Esquema elétrico

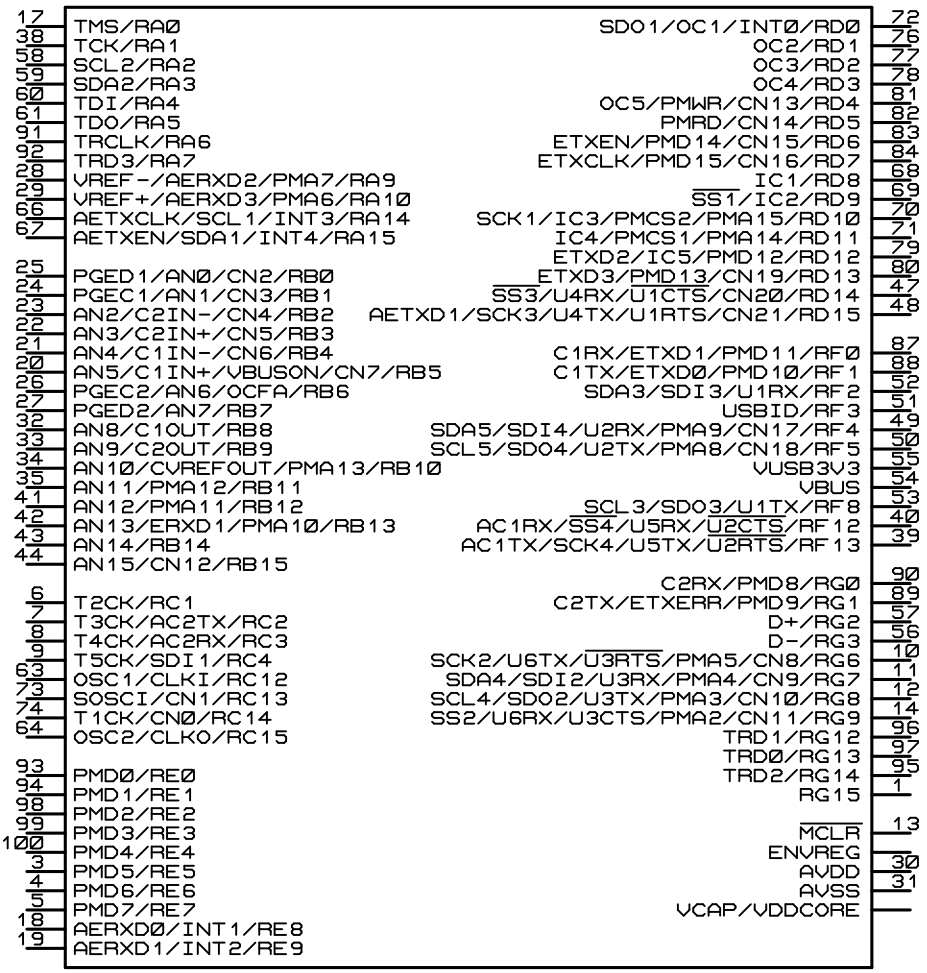

Figura C.1: Esquema dos pinos do microcontrolador PIC32MX795F512L.

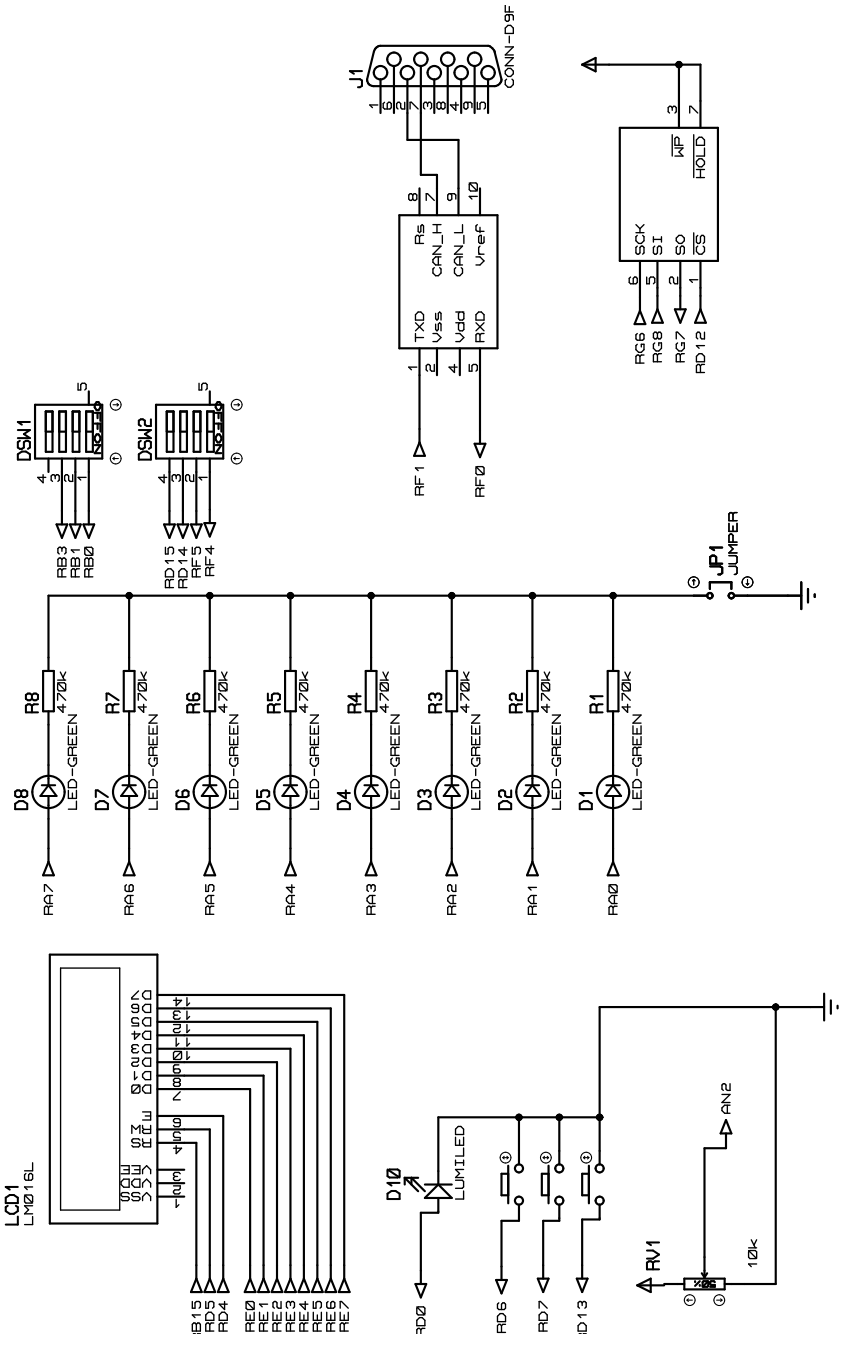

Figura C.2: Esquema elétrico dos nós implementados.